НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

«На правах рукопису»
УДК 004:004.456

«До захисту допущено»

Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2018 р.

Магістерська дисертація
на здобуття ступеню магістра

зі спеціальності 121 – Інженерія програмного забезпечення
(код і назва спеціальності)

на тему: Мобільний додаток обміну захищеними даними з використанням звукових сигналів

Виконав: студент VI курсу, групи ТВ-з71мп

Пономарьов Микола Леонідович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.т.н. Карпенко Є.Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант_____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент ___професор, д.т.н. Кабанячий В.В._____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний
(повна назва)

Кафедра автоматизації проектування енергетичних процесів і систем
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
«___»_____20___ р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Пономарьову Миколі Леонідовичу
(прізвище, ім’я, по батькові)

1. Тема дисертації: Мобільний додаток обміну захищеними даними з використанням звукових сигналів

науковий керівник дисертації Карпенко Євген Юрійович, к.т.н.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « » 2018 р. № _____

2. Термін подання студентом дисертації: 10 грудня 2018 року

3. Об’єкт дослідження: Обмін захищеними даними з використанням звукових сигналів

4. Предмет дослідження: Мобільний додаток обміну даними

5. Перелік питань, які потрібно розробити:

5.1. Аналіз мов програмування для інтегрованого середовища розробки Xcode.
5.2. Аналіз архітектурних шаблонів для розробки мобільного додатку.
5.3 Аналіз технологій обміну даними.
5.4. Аналіз стандартів шифрування даних.
5.5. Аналіз сучасних хмарних СУБД.
5.6. Розробка алгоритму роботи мобільного додатку.
5.7. Створення мобільного додатку.
5.8. Розробка стартап-проекту.

6. Орієнтовний перелік ілюстративного матеріалу:

6.1. Презентація PowerPoint відповідно до теми дисертації.

7. Орієнтовний перелік публікацій:

7.1. Доповідь на науково-технічній конференції з публікацією тез.

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання «____»_____201___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Затвердження теми роботи	до 17.05.2018 р.	
2.	Вивчення та аналіз задачі. Проведення дослідження по вибраній темі	до 03.09.2018 р.	
3.	Розробка архітектури та загальної структури системи	до 28.09.2018 р.	
4.	Програмна реалізація системи	до 26.10.2018 р.	
5.	Захист програмного продукту	до 22.11.2018 р.	
6.	Оформлення пояснювальної записки	до 10.12.2018 р.	
7.	Передзахист	до 30.11.2018 р.	
8.	Захист	з 17.12.2018 р.	

Студент _____ Пономарьов М.Л._____
(підпис) (прізвище та ініціали)

Науковий керівник _____ Карпенко Є.Ю._____
(підпис) (прізвище та ініціали)

Реферат

Загальний обсяг роботи – 117 ст., 27 ілюстрацій, 34 таблиць, 6 додатків, кількість бібліографічних найменувань – 50.

Актуальність теми. Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій в останні роки. Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці та в будь-якій ситуації. Тому важливу роль відіграє чи може користувач оперувати та ділитися даними зі свого пристрою.

Наприклад, одним з безлічі типів додатків для мобільних пристроїв є додатки для подорожей і туризму. Під час таких подорожей вони повинні краще працювати в офлайн-режимі, тому що то не завжди у будь-якому місці та в будь-якій ситуації, у вас буде доступ до інтернету або мобільного оператора.

Мета та задачі дослідження. Метою роботи є розробка мобільного додатку, який використовує звукові сигнали як спосіб обміну даними між користувачами, а також й інші методи за допомогою яких можна здійснювати обмін даними, а саме:

- інтернет;
- QR-код.

Відповідно до мети ставляться такі завдання:

1. Аналіз мов програмування для інтегрованого середовища розробки Xcode.
2. Аналіз архітектурних шаблонів для розробки мобільного додатку.
3. Аналіз технологій обміну даними.
4. Аналіз стандартів шифрування даних.
5. Аналіз сучасних хмарних СУБД.
6. Розробка алгоритму роботи мобільного додатку.
7. Створення мобільного додатку.

Рішення поставлених завдань та досягнуті результати. В роботі розглянуто методи, за допомогою яких можливо здійснювати обін даними. Запропоновано реалізацію мобільного додатку на основі трьох методів обміну даними.

Було реалізовано описаний підхід, в реалізації застосовано патерн

проектування MVC для ефективної роботи мобільного додатку, обробки помилок, пов’язаних в тому числі і з комунікацією між сервером, та іншими пристроями.

Було обрано алгоритм шифрування даних AES128, для забезпечення максимального захисту даних при її передачі.

Реалізацію мобільного додатку було протестовано на багатьох тестових сценаріях, зроблено висновки щодо особливостей нового підходу обміну даними.

Об’єкт досліджень. Об'єктом дослідження є обмін захищеними даними з використанням звукових сигналів.

Предмет досліджень. Предметом дослідження є мобільний додаток з обміну захищеними даними.

Методи досліджень. Для розв’язання зазначеної проблеми в роботі застосовано методи моделювання архітектури мобільного додатку, композиції логічних структур даних, тестування, та логічного узагальнення отриманих результатів.

Практичне значення одержаних результатів. Отриманий результат після реалізації мобільного додатку може використовуватись в інших операційних системах, або веб-сервісах. Представлений приклад реалізації демонструє, що мобільний додаток можна використовувати в сфері подорожей або туризму.

Апробації результатів дисертації. Зроблено доповідь на конференції молодих учених у грудні 2018 р.

Публікації. Пономарьов М.Л. Мобільний додаток обміну захищеними даними з використанням звукових сигналів // Конференція молодих учених — гудень 2018.

Ключові слова: мобільний додаток, iOS, Xcode, шифрування даних, обмін даними, звукові сигнали.

Реферат

Общий объем работы - 117 в., 27 иллюстраций, 34 таблиц, 6 приложений, количество библиографических наименований - 50.

Актуальность темы. Мобильные приложения стали одним из главных трендов в развитии информационных технологий в последние годы. Основным преимуществом мобильных приложений является то, что пользователь может получить доступ к ним, находясь в любом месте и в любой ситуации. Поэтому важную роль играет ли пользователь оперировать и делиться данными с устройства.

Например, одним из множества типов приложений для мобильных устройств являются приложения для путешествий и туризма. Во время таких путешествий они должны лучше работать в офлайн-режиме, потому что то не всегда в любом месте и в любой ситуации, у вас будет доступ к интернету или мобильного оператора.

Цель и задачи исследования. Целью работы является разработка мобильного приложения, которое использует звуковые сигналы как способ обмена данными между пользователями, а также и другие методы с помощью которых можно осуществлять обмен данными, а именно:

- интернет;
- QR-код.

Согласно цели ставятся следующие задачи:

1. Анализ языков программирования для интегрированной среды разработки Xcode.
2. Анализ архитектурных шаблонов для разработки мобильного приложения.
3. Анализ технологий обмена данными.
4. Анализ стандартов шифрования данных.
5. Анализ современных облачных СУБД.
6. Разработка алгоритма работы мобильного приложения.
7. Создание мобильного приложения.

Решение поставленных задач и достигнутых результатах. В работе рассмотрены методы, с помощью которых можно осуществлять Обин данными. Предложена реализация мобильного приложения на основе трех методов обмена данными.

Было реализовано описан подход, в реализации применен паттерн проектирования MVC для эффективной работы мобильного приложения, обработки ошибок, связанных в том числе и с коммуникацией между сервером и другими устройствами.

Был избран алгоритм шифрования данных AES128, для обеспечения максимальной защиты данных при их передаче.

Реализацию мобильного приложения было протестировано на многих тестовых сценариях, сделаны выводы об особенностях нового подхода обмена данными.

Объект исследований. Объектом исследования является обмен защищенными данными с использованием звуковых сигналов.

Предмет исследований. Предметом исследования является мобильное приложение по обмену защищенными данными.

Методы исследований. Для решения указанной проблемы в работе применены методы моделирования архитектуры мобильного приложения, композиции логических структур данных, тестирование и логического обобщения полученных результатов.

Практическое значение полученных результатов. Полученный результат после реализации мобильного приложения может использоваться в других операционных системах, или веб-сервисах. Представлен пример реализации показывает, что мобильное приложение можно использовать в сфере путешествий или туризма.

Апробации результатов диссертации. Сделан доклад на конференции молодых ученых в декабре 2018

Публикации. Пономарёв Н.Л. Мобильное приложение обмена защищенными данными с использованием звуковых сигналов // Конференция молодых ученых - декабрь 2018.

Ключевые слова: мобильное приложение, iOS, Xcode, шифрование данных, обмен данными, звуковые сигналы.

Abstract

Total volume of work - 117 items, 27 illustrations, 34 tables, 6 applications, number of bibliographic titles - 50.

Actuality of theme. Mobile applications have become one of the main trends in the development of information technology in recent years. The main advantage of mobile apps is that the user can access them, while in any place and in any situation. Therefore, an important role is played by the user can operate and share data from his device.

For example, one of the many types of mobile apps is travel and tourism applications. During such trips, they should work better offline, because they are not always in any place and in any situation, you will have access to the Internet or mobile operator.

Purpose and tasks of the research. The aim of the work is to develop a mobile application that uses audio signals as a way of exchanging data among users, as well as other methods by which data can be exchanged, namely:

- internet;
- QR code.

According to the goals are the following tasks:

1. Analysis of programming languages for the integrated development environment Xcode.
2. Analysis of architectural templates for the development of a mobile application.
3. Analysis of data exchange technologies.
4. Analysis of data encryption standards.
5. Analysis of modern cloud DBMS.
6. Development of the algorithm of mobile application.
7. Create a mobile application.

Solution of the tasks and achieved results. The paper deals with the methods by which it is possible to carry out data obing. The implementation of a mobile application based on three methods of data exchange is proposed.

The described approach was implemented, MVC design pattern for the effective operation of the mobile application, handling of errors, including communication with the server, and other devices were implemented in implementation.

The AES128 data encryption algorithm was chosen to provide maximum data protection when it was transmitted.

The implementation of the mobile application has been tested on many test scenarios, conclusions have been made regarding the features of the new data exchange approach.

Object of research. The object of the study is the exchange of protected data using sound signals.

Subject of research. The subject of the study is a mobile application for the exchange of protected data.

Research methods. To solve this problem, the methods of modeling the architecture of the mobile application, the composition of the logical data structures, testing, and the logical generalization of the obtained results are used in the work.

The practical value of the results. The resulting result after the implementation of the mobile application can be used on other operating systems or web services. The presented example of implementation demonstrates that the mobile application can be used in the field of travel or tourism.

Approbation of the results of the dissertation. A report was presented at a conference of young scientists in December 2018.

Publications. Ponomarov M.L. Mobile application for the exchange of protected data using sound signals // Conference of Young Scientists - December 2018.

Keywords: mobile app, iOS, Xcode, data encryption, data exchange, audio signals.

ЗМІСТ

Перелік умовних позначень

СУБД – Система управління базами даних

ООП – Об'єктно-орієнтоване програмування

MVC – Model View Controller

MVP – Model View Presenter

MVVM – Model View View-Model

VIPER – View Interactor Presenter Entities Router

UI – User Interface

API – Application Programming Interface

ARPANET – Advanced Research Projects Agency Network

ICANN – Internet Corporation for Assigned Names and Numbers

IETF – Internet Engineering Task Force

DES – Data Encryption Standard

AES – Advanced Encryption Standard

FEAL – Fast Enciphering Algoritm

IDEA – International Data Encryption Algorithm

EC2 – Elastic Compute Cloud

AMI – Amazon Machine Image

БД – База Даних

Вступ. Актуальність теми та її практичне значення

Актуальність теми. Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій в останні роки. Кількість розробників мобільних додатків збільшується, кількість доступних додатків зростає. Все більше компаній зацікавлені в розробці програми, яка допоможе їм досягти успіху в своїй галузі і обійти конкурентів.

Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці і в будь-якій ситуації. Це відкриває великі можливості щодо впровадження різних сервісів, які раніше були недоступні. Важливу роль також відіграє соціалізація додатків, тобто додавання функцій соціальних мереж, які дозволяють користувачем ділитися своїми досягненнями і знаходити нових друзів.

Одним з безлічі типів додатків для мобільних пристроїв є додатки для подорожей і туризму. Особливо під час таких подорожей мобільні додатки повинні краще працювати в оффлайн режимі, наприклад у ті моменти коли мобільний пристрій не має сигналу оператора мобільного зв’язку, або доступу до мережі інтернет. На даний момент багато туристів мають мобільні пристрої і використовують їх під час подорожей, і компанії, що не мають таких додатків, втрачають безліч потенційних клієнтів.

Мета та задачі дослідження. Метою роботи є розробка мобільного додатку, який використовує звукові сигнали як спосіб обміну даними між користувачами, а також й інші методи за допомогою яких можна здійснювати обмін даними, а саме:

- інтернет;
- QR-код.

Відповідно до мети ставляться такі завдання:

1. Аналіз мов програмування для інтегрованого середовища розробки Xcode.
2. Аналіз архітектурних шаблонів для розробки мобільного додатку.
3. Аналіз технологій обміну даними.
4. Аналіз стандартів шифрування даних.
5. Аналіз сучасних хмарних СУБД.
6. Розробка алгоритму роботи мобільного додатку.
7. Створення мобільного додатку.

Рішення поставлених завдань та досягнуті результати. В роботі розглянуто методи, за допомогою яких можливо здійснювати обмін даними. Запропоновано реалізацію мобільного додатку на основі трьох методів обміну даними.

Було реалізовано описаний підхід, в реалізації застосовано патерн проектування MVC для ефективної роботи мобільного додатку, обробки помилок, пов’язаних в тому числі і з комунікацією між сервером, та іншими пристроями.

Було обрано алгоритм шифрування даних AES128, для забезпечення максимального захисту даних при її передачі.

Реалізацію мобільного додатку було протестовано на багатьох тестових сценаріях, зроблено висновки щодо особливостей нового підходу обміну даними.

Об’єкт досліджень. Об'єктом дослідження є обмін захищеними даними з використанням звукових сигналів.

Предмет досліджень. Предметом дослідження є мобільний додаток з обміну захищеними даними.

Методи досліджень. Для розв’язання зазначеної проблеми в роботі застосовано методи моделювання архітектури мобільного додатку,

композиції логічних структур даних, тестування, та логічного узагальнення отриманих результатів.

Практичне значення одержаних результатів. Отриманий результат після реалізації мобільного додатку може використовуватись в інших операційних системах, або веб-сервісах. Представлений приклад реалізації демонструє, що мобільний додаток можна використовувати в сфері подорожей або туризму.

Апробації результатів дисертації. Зроблено доповідь на конференції молодих учених у грудні 2018 р.

Публікації. Пономарьов М.Л. Мобільний додаток обміну захищеними даними з використанням звукових сигналів // Конференція молодих учених — гудень 2018.

Структура роботи

Дисертація складається зі вступу, 7 розділів, загальних висновків, списку використаних джерел із 50 найменувань.

Основний текст дисертації викладено на 117 стор.

РОЗДІЛ 1

АНАЛІЗ МОВ ПРОГРАМУВАННЯ ДЛЯ ІНТЕГРОВАНОГО СЕРЕДОВИЩА РОЗРОБКИ XCODE

1.1 Мова програмування Objective C

Objective-C — це високорівнева об'єктно-орієнтована мова програмування загального призначення, розроблена як набір розширень стандартної C[1].

Мова програмування Objective-C була розроблена на початку 1980-х років. Вона була обрана в якості основної мови, яка використовувалася компанією «NeXT» для операційної системи (ОС) NeXTSTEP, на базі якої створили macOS і iOS [1].

На цей час вона використовується в у macOS та GNUStep — середовищах, які були розроблені на основі OpenStep стандарту, та Cocoa — це бібліотека компонентів для розробки різних програм. Будь яку програму написану на Objective-C що не використовує Cocoa та OpenStep можна скомпілювати для будь-якої іншої платформи, яку підтримує gcc компілятор та Objective-C.

Objective-C є розширенням мови програмування C і тому компілятором Objective-C може скомпілювати будь-яку програму написану на C [1].

ООП в Objective-C включає інтерфейси, класи, категорії. Реалізовано одиничне, невіртуальне спадкування, також немає єдиного базового класу для всіх об'єктів. Всі методи в класі — є віртуальними, а категорія — є парадигмою яка дозволяє описувати інтерфейс з методами які «необов'язково» створювати [1].

C та Smalltalk одночасно є тими технологіями звідки пішов синтаксис мови Objective-C. Від Smalltalk взято основний семантичний конструкт мови — це замість виклику методу, ми надсилаємо повідомлення об’єкту. Наприклад,

якщо obj як клас об'єкта містить метод doJob то потрібно говорити що об'єкт класу obj відкликається на повідомлення doJob. Для того щоб надіслати повідомлення doJob потрібно написати:

[obj doJob];

Це дозволяє нам надсилати повідомлення навіть до тих об'єктів які не підтримують їх обробки. Такий підхід відрізняється від тих що використовуються в статично типізованих мовах C++ чи Swift [1].

У мові програмування Objective-C для позначення об'єктів використовується спеціальний тип id, тому змінна типу id фактично є вказівником на довільний об'єкт наприклад типу void в мові C. Для позначення нульового вказівника на об'єкт використовується константа nil що дорівнює NULL в мові C. Також замість id загалом можна використовувати і більш звичні позначення, такі як з явною вказівкою класу. В свою чергу якщо ми явно вказуємо клас то компілятор здійснює деяку перевірку підтримки повідомлення об'єктом - якщо компілятор з не може зробити висновок про підтримку цим об'єктом даного повідомлення, то він видає попередження. Тим самим мова Objective-C підтримує перевірку типів в нестрогій формі. Тобто знайдені невідповідності повертаються як попередження, а не помилки [1]. Для відправлення повідомлень або звернень використовується наступний синтаксис:

[receiver message];

У цій конструкції receiver є вказівником на будь який об'єкт, а message – це ім’я методу. Порівнюючи з C++, в Objective-C під час надсилання nil'у як повідомлення - це є дозволеною операцією, яка завжди повертає nil, тобто нульове значення. Приклад передавання параметрів:

[myRect setOrigin:30.0 :50.0];

У прикладі іменем методу є setOrigin. Звертаючи увагу, що для кожного аргументу спочатку ставиться одна двокрака. При цьому в наведеному прикладі перший аргумент має ім’я – це текст перед двокрапкою, а другий - ні. Objective-C дозволяє забезпечувати ім’ям кожен аргумент, що звичайно підвищує

читаність коду і знижує ймовірність передачі неправильного параметра. Саме такий стиль прийнятий більшістю розробників [1].

```
[myRect setWidth:10.0 height:20.0];
```

У цьому прикладі іменем повідомлення виступає setWidth: height:. Також підтримується можливість передачі довільної кількості аргументів в повідомленні:

```
[myObject makeGroup: obj1, obj2, obj3, obj4, nil];
```

Як і функції, повідомлення можуть повертати значення. Але на відміну від мови програмування С, тип значення, яке повертається, є id.

```
float area = [myRect area];
```

Приклад як результат одного методу можна відразу ж використовувати як аргумент в іншому повідомленні:

```
[myRect setColor:[otherRect color]];
```

Як зазначалося, в Objective-C класи це і є об'єкти. Головним завданням таких об'єктів, так званих class objects, є створення екземплярів даного класу, при цьому ім'я цього класу відіграє подвійну роль – по перше, воно виступає типом даних як в мові С, тобто ім’я використовують для опису вказівників на об'єкти даного класу, по друге - ім'я класу може виступати об'єктом, якому надсилається повідомлення, наприклад для ініціалізації [1].

```
Rect * myRect = [[Rect alloc] init];
```

Загалом у мові Objective-C немає вбудованого типу для булевих величин, тому. Для логічних величин в операційних системах NextStep, Mac OS X використовуть тип BOOL з можливими значеннями YES і NO. Першим досить масштабним застосуванням мови Objective-C було в операційній системі NextStep. Для ОС було написано величезну кількість різних класів на Objective-C, багато з яких зараз використовують в Mac OS X. У всіх імен цих класів починаються з префікса NS, що і показує те що вони належать до операційної системи NextStep. Зараз ці класи входять в бібліотеку Foundation, на якій створюються програми для macOS, iOS, WatchOS, tvOS. Наприклад – NSString. Цей клас працює з рядками, при тому що для внутрішнього представлення

символів використовується Юнікод. Компілятор автоматично підтримує даний тип, наприклад він переводить конструкції виду @ "my string" у вказівник на об'єкт класу NSString, який містить дані з цього рядка.

Властивості

Припустимо, в класі Company існує instance-змінна name.

```
@interface Company : NSObject {
    NSString *name;
}
```

Для доступу до цього рядку ззовні найкраще скористатися властивостями, які з'явилися у Objective C 2.0. Цей рядок зможуть побачити усі інші об’єкти які будуть унаслідуватись від класу Company. Для декларації властивостей об’єкту використовується ключове слово @ property.

```
@property (retain) NSString *name;
```

У дужках перераховуються атрибути доступу до instance-змінної. Атрибути поділяються на 3 основні групи.

Імена акцессора і мутатора

getter = getterName, це ім’я використовується для задання імені функції, яка використовується для отримання значення цієї instance-змінної.

setter = setterName, це ім’я використовується для задання імені функції, яка використовується для задання значення цієї instance-змінної.

Обмеження читання / запису

readwrite - властивість може бути "прочитана" та перезаписана

readonly - властивість може бути тільки "прочитана"

Ці атрибути взаємо виключають один одного.

Атрибути мутатора.

assign - для задання нового значення використовується оператор присвоєння. Цей атрибут використовується тільки для POD-типів або для об'єктів, якими ми не володіємо [1].

retain – як атрибут вказує на те, що цьому об'єкту, який використовується як нове значення instance-змінної, управління пам'яттю буде відбуватися вручну, тому не потрібно забувати потім звільнити пам'ять [1].

copy - вказує на те, копія переданого об’єкту буде використана для присвоєння[1.

weak - аналог assign при застосуванні режиму автоматичного підрахунку посилань. (ARC повинен підтримуватися компілятором)

strong - аналог retain при застосуванні режиму автоматичного підрахунку посилань . (ARC повинен підтримуватися компілятором)

При роботі під GC ми не маємо ніякої різниці між використанням assign, retain, copy. Для створення властивостей коду, згідно з декларацією, можна скористатися автогенерацією коду, наприклад:

@synthesize name;

Автоматично створений код - це не завжди оптимальне рішення і може знадобитися створення методів доступу до instance-змінних вручну. Objective-C часто критикують за перевантажений синтаксис, в порівнянні з іншими мовами. Однак при цьому частіше кажуть про те що читаність цієї мови дуже велика [1].

1.2 Мова програмування Swift

Swift – відкрита мультипарадигмальна компілюєма мова програмування загального призначення. Створено компанією Apple в першу чергу для розробників iOS і macOS. Swift працює з фреймворками Cocoa і Cocoa Touch і сумісний з основною кодовою базою Apple, написаної на Objective-C. Свіфт задумався як більш легкий для читання і стійкий до помилки програміста мови, ніж предшествовавший йому Objective-C. Програми на Swift компілюються за допомогою LLVM, входячи в інтегровану середовище розробки Xcode. Swift може використовувати об'єкт Objective-C, що робить можливим використання обох мов (а також C) в межах однієї програми.

1.2.1 Історія

Старший віце-президент з розробки програмного забезпечення Apple Craig Fidderigy під час анонса цього продукту заявив, що програма мови Swift була закладена ще в NeXT платформі (OS NeXTSTEP яка випускалася в 1989-1995 роках), яка стала основою для сучасної MacOS, а потім і iOS [2].

Розробка поточного варіанту Swift почалася в 2010 році Крисом Латтнером, керівником відділу розробки інструментів для створення програмного забезпечення Apple та одним з основних розробників LLVM. Свіфт запозичив ідеї з «Objective-C, Rust, Haskell, Ruby, Python, C #, CLU, і ще з багатьох сторінок, що складно перечислити» [2].

2 червня 2014 року на конференції WWDC Swift була офіційно представлена разом з безкоштовним керівництвом з використанням обсягом в 500 сторінок, доступним на сайті «iBook Store».

8 июня 2015 года компанія Apple оголосила про випуск нової версії Swift 2.0, яка отримала більш високу продуктивність, нову обробку помилок API, поліпшенню синтезу мови, а також функції перевірки доступності функцій Swift для цільових ОС [2].

3 грудня 2015 року була випущена бета версія Swift 3.0 з підтримкою операційних систем OS X, iOS і Linux та ліцензірованной під відкритою ліцензією ліцензії Apache 2.0 з винятком бібліотеки Runtime. Версія 3.0 назад не сумісна з більш ранніми версіями мови; начиная з нативної середовища розробки XCode версії 9 більш версії мови Swift-2 і раніше не підтримуються.

На початку квітня 2016 року невідомий джерело інформації в корпорації Google повідомив, що компанія розглядає можливість перекладу Swift мови в "перший клас" для платформи Android. Раніше вже предъявлялись прототипы Swift компилятора для Android.

19 вересня 2017 року була випущена версія Swift 4.0.

У вересні 2018 року, разом з новою версією iOS 12, була випущена нова версія мови Swift 4.2. В якій заявлено, нарешті, стабільна робота ABI з

стандартними бібліотеками (Swift Dynamic Library), підтримка регулярних виразів і першокласне рішення для паралельної обробки даних з асинхронним режимом обробки async / очікування [2].

1.2.2 Опис мови Swift

Swift запозичив досить багато з Objective-C, проте він визначається не покажчиками, а типами змінних, які обробляє компілятор. За аналогічним принципом працюють багато скриптові мови. У той же час, він надає розробникам багато функцій, які раніше були доступні в C ++ і Java, такі як визначаються найменування, так звані узагальнення і перевантаження операторів.

Частина функцій мови виконується швидше в порівнянні з іншими подібними мовами. Наприклад, сортування комплексних об'єктів виконується в 8,4 разів швидше, ніж в Python, і майже в 2,6 рази швидше, ніж в Objective-C [3].

Код, написаний на Swift, може працювати разом з кодом, написаним на мовах програмування C і Objective-C в рамках одного і того ж проекту. В програмуванні C і Objective-C в рамках одного й того самого проекту [2].

1.2.3 Приклади синтаксису мови Swift

Для створення константи використовуйте let, а для створення змінної var. Для константи вказувати тип не потрібно, ви можете присвоїти значення лише один раз[4].

```
var myVariable = 42
myVariable = 50
let myConstant = 42
```

Типи змінних та констант повинні збігатися з тими типами які привласнюються їм з відповідних значень. Однак не потрібно безпосередньо вказувати їх тип. Компілятор автоматично розуміє та визначає тип константи і змінної при присвоєнні їм значення. У наведеному прикладі компілятор визначить, що змінна myVariable має цілочисельний тип [4].

Якщо ж ініціалізатор відсутній або інформацію яку він надає не достатньо, ви можете самостійно вказати тип після змінної, розділивши двокрапкою назву і тип:

```
let implicitInteger = 70
let inplicitDouble = 70.0
let inplicitDouble: Double = 70
```

Значення ніколи не конвертуються в інший тип неявно. Якщо необхідно конвертувати значення в будь-який інший тип, ось приклад явного конвертування:

```
let label = "The width is"
let width = 94
let widthLabel = label + String (width)
```

Є простіший спосіб включення значень в рядки: для цього укладіть вираз в дужки і поставте перед ними зворотний слеш (\). приклад:

```
let apples = 3
let oranges = 5
let appleSummary = "I have \ (apples) apples."
let fruitSummary = "I have \ (apples + oranges) pieces of fruit."
```

При роботі з словниками, масивами (ще словники називають асоціативними масивами) використовуються квадратні дужки []:

```
var shoppingList = ["catfish", "water", "tulips", "blue paint"]
shoppingList [1] = "bottle of water"
```

```
var occupations = [
    "Malcolm": "Captain",
    "Kaylee": "Mechanic",
]
```

Щоб створити порожній масив або dictionary, використовується наступний синтаксис:

```
let emptyArray = String [] ()
let emptyDictionary = Dictionary <String, Float> ()
```

Для того щоб створити умови використовуються оператори if і switch, for-in, for, while і do-while - використовують для створення циклів. При цьому виділяти круглими дужками умови і ініціалізувати вираження необов'язково, тоді як фігурні дужки обов'язкові [4].

```
let individualScores = [75, 43, 103, 87, 12]
var teamScore = 0
for score in individualScores
{
    if score> 50 {
        teamScore + = 3
    } Else {
        teamScore + = 1
    }
}
```

Висновок до розділу

В даному розділі був проведений аналіз двух мов програмування, Objective C та Swift. Було розглянуто синтаксис, а також сильні та слабкі сторони при використанні цих технологій.

Сильною стороною мови програмування Objective C є те що ця мова існує вже більше 20-ти років, тому вона вважається дуже стабільною та опрацьованою. Існує безліч інформації та прикладів реалізації питань, що можуть виникати під час використання цієї технології під час розробки. Сильною стороною також вважається те що цю технологію можна використовувати й з іншими мовами, такими як Objective C++, C, C++ та Swift. Недоліком можна вважати те що ця мова поступається швидкістю компіцяції з більш новою мовою програмування для платформи iOS – Swift.

Сильною стороною мови програмування Swift є те що ця мова дуже швидка якщо порівняти її з Objective C. Також цю технологію можна використовувати й з іншими мовами, такими як Objective C++, C, C++ та Objective C. Але головним недоліком є те що ця мова існує тільки 4 роки, тому вона знаходиться ще в стадії розробки та доопрацювання. Тому можна вважати що використання цієї мови для розробки мобільно додатку не є доцільним.

РОЗДІЛ 2

АНАЛІЗ АРХІТЕКТУРНИХ ШАБЛОНІВ ДЛЯ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

Існує безліч різних архітектурних шаблонів для розробки мобільних IOS додатків. Вибір архітектури майбутнього додатки дуже важливий етап проектування. Правильний вибір архітектури, дозволить проводити пошук помилок, а так само оперативно реагувати на зміни всередині класу. В даний час є безліч варіантів шаблонів проектування архітектури:

- MVC;
- MVP;
- MVVM;
- VIPER.

MVC, MVP, MVVM припускають включення об'єктів додатки в одну з трьох категорій:

- Models - відповідальні за дані домену або рівень доступу до даними, який маніпулює даними;
- Views - відповідальні за рівень представлення (Graphics User Interface), всі елементи з префіксом "UI";
- Controller, Presenter, ViewModel - виконує роль посередника між Model і View, що відповідає за зміни Model. Реагуючи на дії користувача, що виконуються під View і оновлюючи View зі змінами з Model.

Далі докладніше розглянемо кожен із наведених вище шаблонів.

2.1 Традиційний MVC

При використанні шаблону традиційного MVC, View не має стану, він просто відображається контролером після зміни моделі [10]. Таким чином, цей шаблон на увазі повне перезавантаження сторінки після будь-якої дії користувача. Таким чином, реалізація традиційного MVC в додатку IOS НЕ має сенсу через архітектурної проблеми - все три об'єкта пов'язані один з другом і кожна сутність знає про двох інших.

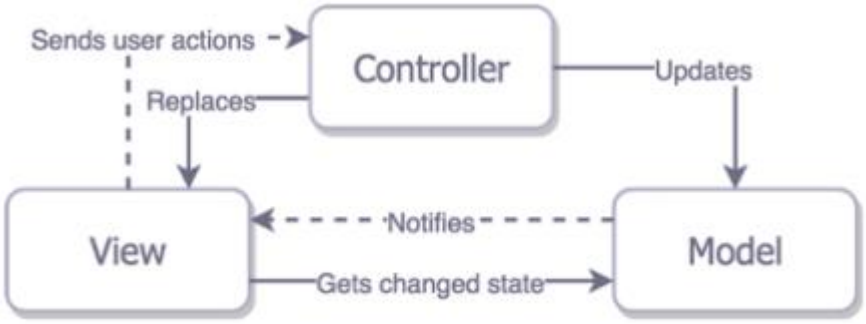


Рис. 2.1 - Традиційний MVC

У шаблони архітектури Apple MVC контролер буде посередником між View і Model, що б ці два об'єкти не знали про існування один одного [11]. Найбільш часто використовуваним об'єктом буде бути контролер. У цьому є плюси, тому що в додатку повинно бути місце для всієї складної бізнес-логіки, яка не вписується в Model.

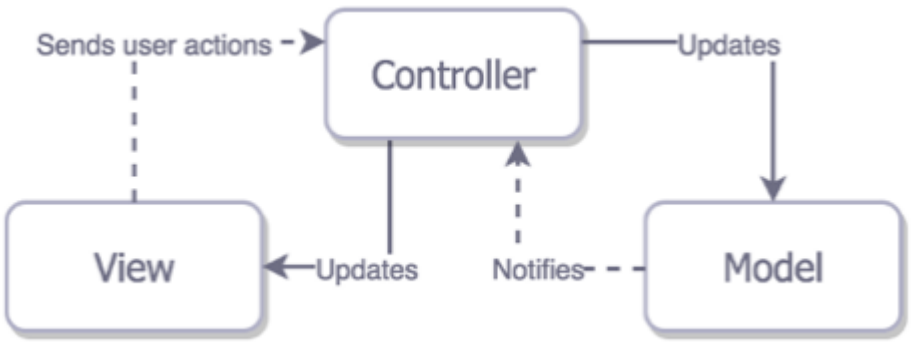


Рис. 2.2 – Apple MVC

2.2. Практичний Apple MVC

В реальності шаблон Apple MVC працює по-іншому. Перевантажений контролер містить в собі велику кількість бізнеслогікі, перетворюючи Model View-Controller в Massive-View-Controller. Cocoa MVC заохочує писати Massive-View-Controller, тому що вони дуже сильно залучені в життєвий цикл View, що призводить до нероздільності цих двох сутностей.

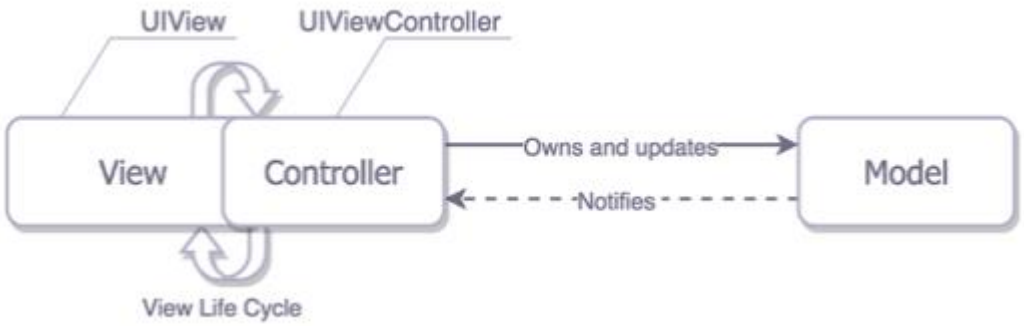


Рис. 2.3 – Практичний Apple MVC

2.3 Шаблон MVP

MVP (Cocoa MVC's promises delivered) схожий на Apple MVC, але це не так [12]. В Apple MVC - View пов'язаний з контролером, в той час як медіатор MVP - Presenter, не має нічого спільного з життєвим циклом контролера, тому в Presenter немає коду для установки макета екрану, але він відповідає за оновлення View з актуальними даними і станом. MVP це архітектурний шаблон, який показує проблеми збірки, яка виникає через наявність трьох фактично розділених шарів.

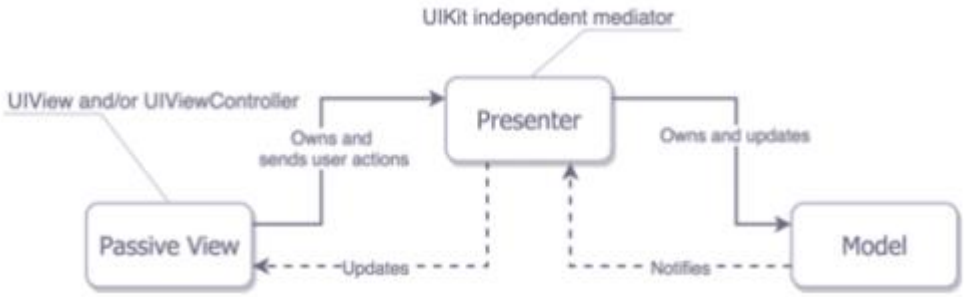


Рис. 2.4 – MVP

2.4 Інший формат шаблону MVP

Існує й інший формат MVP - MVP Supervising Controller [12]. Цей варіант включає пряму залежність View і Model, в той час як Presenter (Supervising Controller) все ще обробляє дії з View і здатний його змінювати. Поділ невиразною відповідальності, є неприпустимим, так само як і сильна зв'язок View і Model.

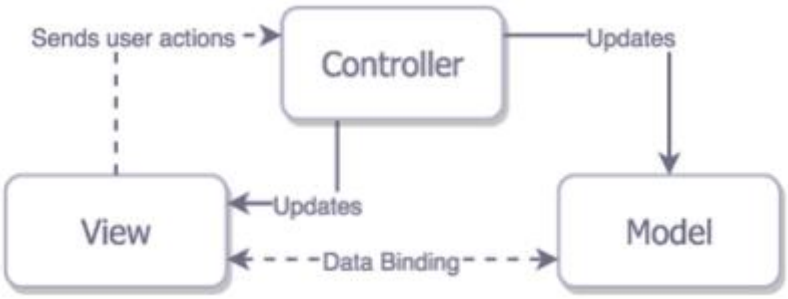


Рис. 2.5 - MVP (With Bindings and Hooters)

2.5 Шаблон MVVM

MVVM розглядає контролер View, як View, а так само має на увазі відсутність зв'язку між View і Model [13]. View-Model забезпечує незалежне уявлення View і його стану. View-Model викликає зміни в Model і оновлює себе

з оновленою Model, а оскільки View і View-Model з'єднані, View відповідно теж оновлюється.

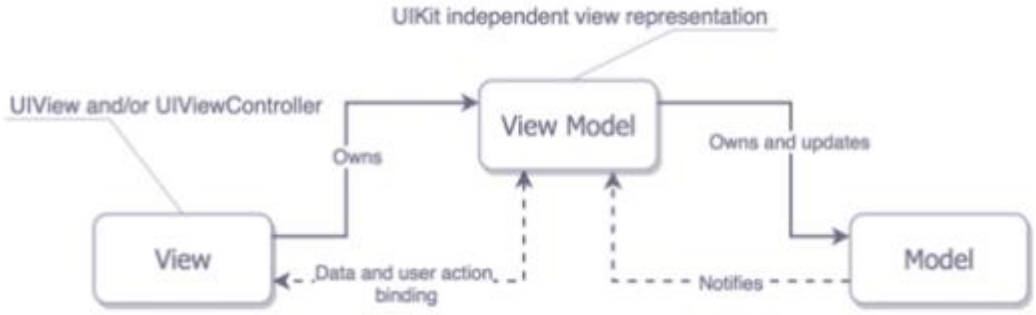


Рис. 2.6 – MVVM

2.6. Шаблон VIPER

VIPER дозволяє повністю розділити обов'язки, і забезпечити повну незалежність модулів [14]. Він складається з:

- View - відповідає за виконання сталися в Presenter змін;
- Interactor - містить бізнес-логіку, пов'язану з даними (Entities) або мережевими пристроями. У його обов'язки входять такі операції, як створення нових екземплярів об'єктів, або вибірку їх з сервера. Для цих цілей використовуються служби і менеджери, які не є частиною модуля VIPER - вони є зовнішньої залежністю;
- Presenter - містить пов'язану з UI (незалежну від UIKit) бізнес-логіку, викликає методи в Interactor;
- Entities - в цьому класі зберігаються прості об'єкти даних;
- Router - відповідає за переходи між різними модулями VIPER.

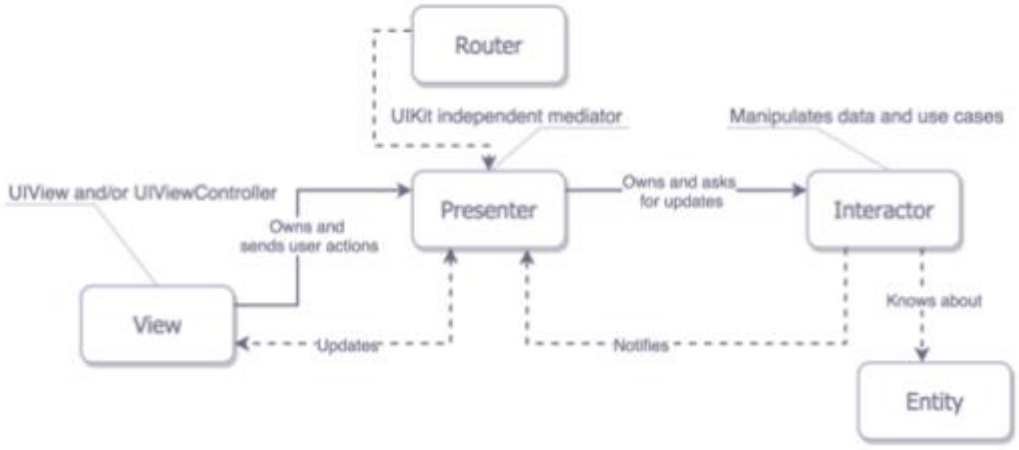


Рис. 2.7 – VIPER

Модуль VIPER може відповідати як одному екрану в додатку, так і всьому додатком, чітких вказівок як вибрати розмір модуля не наводиться. VIPER - це перший шаблон в якому явно розглядається навігаційна відповідальність, яка вирішується за допомогою Router.

Висновок по розділу

В результаті аналізу архітектурних шаблонів були зроблені наступні висновки:

1) Традиційний MVC - не застосовний до сучасної IOS розробці.
2) Apple MVC є найкращим архітектурним шаблоном, з точки зору швидкості розробки, але помітно ускладнює підтримку додатка при його збільшенні.
3) MVP пропонує хорошу тестованого, а й значне збільшення коду. Його особливості полягають в розподіл - велика частина обов'язків, розділені між Presenter і Model, при цьому у View залишається мінімум завдань. Так само перевагою є тестованого - можна протестувати більшу частину бізнес-логіки через практичних порожнього View. MVP, як же як і традиційний MVC є неприпустимим для використання в IOS розробці.

4) Особливості MVVM полягають в поширенні - View в MVVM має більше обов'язків, ніж View в MVP, оскільки MVVM оновлює стан з ViewModel, а другий тільки передає всі події в Presenter і не оновлює себе. У тестованості - ViewModel нічого не знає про View що дозволяє без проблем його тестувати. View також може бути протестований, але оскільки він залежить від UIKit - тестування його НЕ доцільно. Таким чином, MVVM поєднує в собі переваги всіх розглянутих раніше підходів до архітектури IOS додатків, в той же час не вимагає додаткового написання коду для поновлення View. Тестованих так само знаходиться на хорошому рівні;
5) Viper перевершує всі представлені архітектурні патерни в розподіленої, тестованості, і простоті освоєння нових модулів. Основними особливостями Viper є поширення - VIPER відрізняється відмінною здатністю до розподілу обов'язків, тестованого - при кращому розподілі, виходить найкраща тестованого. Так само зручність використання - невелика кількість коду в блоках VIPER, дозволяє без зусиль перемикатися з одного завдання на інше, збільшуючи тим самим корисну роботу програміста.

Ураховуючи те що наразі наш проект є лише прикладом для використання звукових сигналів, можна зробити висновок, що для найкращого розподілу обов'язків, і покриття додатки тестами, найбільше підходить архітектурний шаблон – практичний Apple MVC.

РОЗДІЛ 3

АНАЛІЗ ТЕХНОЛОГІЙ ОБМІНУ ДАНИМИ

3.1 Інтернет

Інтернéт (від англ. Internet),— це всесвітня система сполучених між собою комп'ютерних мереж, що базуються на Інтернет-протоколах. Інтернет також можна назвати мережею мереж. Він складається з мільйонів публічних і урядових, академічних, локальних, приватних, ділових і глобальних мереж, пов'язаних між собою з використанням дуже різноманітних бездротових, оптичних і дротових технологій. Інтернет становить є великою фіщичною основою для розміщення величезної кількості інформації та ресурсів і послуг, наприклад як взаємопов'язані гіпертекстові документи Всесвітньої павутини або World Wide Web — WWW, та електронна пошта [5].

Зараз слово Інтернет найчастіше вживається в значенні Всесвітньої павутини та інформації яка в ній доступна, а не у значенні мережі як фізичного об’єкту. Також наразі люди вживають й інші терміни, наприклад «Всесвітня мережа», «Глобальна мережа» чи навіть одне слово «Мережа» чи «Інет». Також зараз все частіше Інтернет як слово вживається і з малої літери, що можна пояснити з термінами такі як «радіо», «телебачення», які пишуть з малої [5].

Історія Інтернету як мережі обміну інформацією починається з досліджень на замовлення уряду США 1960-х років, і мали на меті створення надійних та захищених розподілених комп'ютерних мереж, стійких до пошкоджень. До Інтернету першою мережею стала ARPANET з англійскої - Advanced Research Projects Agency Network, яка, наприкінці 1960-х років, на початку 1970-х років об'єднувала близько 200 вузлів [5].

Фінансування магістральної мережі урядом, Національного наукового фонду США в 1980-х роках, а також приватне фінансування для інших

комерційних магістральних мереж в усьому світі сприяло до участі в розробці нових мережевих технологій і злиття багатьох мереж. Комерціалізація яка відбулася в 1990-х міжнародної мережі привела до її популяризації використання та впровадження в практично кожен аспект сучасного життя людини для обміну інформацією. Наразі 2011 року понад 2,1 мільярда людей користуються послугами Інтернету [5].

Інтернет не має централізованого управління, правил використання чи доступу, тому кожна складова мережа встановлює свої стандарти. Наразі централізовано визначаються правила використання адресного простору Інтернет-протоколу та Системи доменних імен. Цим кирує Інтернет-корпорація з присвоєння імен та номерів, міжнародна некомерційна організація з головним офісом у США, з англійської Internet Corporation for Assigned Names and Numbers, або ICANN. Також Internet Engineering Task Force (IETF) - проводить технічне обґрунтування і стандартизацію основних протоколів (IPv4 та IPv6). Це некомерційна організація, являє собою відкриту міжнародну спільнота учених, проектувальників, мережевих операторів і постачальників послуг [5].

Інтернет побудована на маршрутизації пакетів даних і використанні протоколу IP. В наш час ця мережа слугує фізичною основою доступу до веб-сайтів і багатьох систем (протоколів) передачі даних, а також відіграє важливу роль у створенні інформаційного простору глобального суспільства[5].

Для узгодженого прийняття й передавання різних типів даних у мережах існують мережеві протоколи.

Мережевий протокол — це сукупність стандартів або іншими словами - правил для обміну даними між комп’ютерами [6].

Таблиця 3.1

Протоколи та їхнє призначення

Назва протоколу	Призначення протоколу
Ethernet	Стандарт дротових локальних мереж
Wi-Fi	Стандарт бездротових мереж
WAP	Стандарт доступу до мережі з мобільних телефонів
IP	Передавання пакетів даних
TCP	Управління передаванням і цілісністю пакетів даних
DNS	Перетворення доменних імен в Р-адреси
FTP	Обмін файлами між комп'ютерами
HTTP	Передавання гіпертексту
POP	Отримання повідомлень від поштового сервера
SMTP	Відправлення повідомлень на поштовий сервер

3.2 QR-код

QR-код (англ. Quick Response Code - код швидкого реагування; скор. QR code) - товарний знак для типу матричних штрих-кодів (або двовимірних штрих-кодів), спочатку розроблених для автомобільної промисловості Японії. Штрихкод – містить містить інформацію про об'єкт, до якого вона прив'язана, тому при зчитуванні машиною це є оптичною міткою. QR-код використовує чотири стандартизованих режиму кодування (числовий, буквено-цифровий, двійковий і кандзі) для ефективного зберігання даних; можуть також використовуватися розширення [7].

QR-код являє собою квадрат білого фону, який складається з чорних квадратів, розташованих у квадратної сітці. Вони можуть зчитуватися за допомогою пристроїв, наприклад мобільних телефоній, які мають інтегровані

пристрої обробки зображень, таких як камера, і оброблятися з використанням кодів Ріда - Соломона до тих пір, поки зображення не буде розпізнано належним чином, да розшифровано. Потім необхідні дані витягуються з шаблонів, які присутні в горизонтальних і вертикальних компонентах зображення [7].

Також потрібно звернути увагу на те, що чим більший QR-код, тобто кількість даних які були закодовані не велика, тим вірогідніше що цей код він буде зчитуватися коректніше. Однак зараз, враховуючи нові технології зчитування, за допомогою більшості сучасних пристроїв, можна сканувати зображення, які мають досить маленький формат, наприклад, візитну картку. Звичайно, передбачається, що якість зображення QR – коду та параметри камери пристрою яку використовують мають бути хорошими [8].

Наразі використання QR – коду, є доцільним. Тому що, що його можливо використовувати для передачі будь-яких даних типів даних після кодування за допомогою кодів Ріда-Соломона. Серед прикладів можна навести й такі як: звичайні текстові дані, телефонні дані, місцезнаходження на карті, вебсторінки [8].

3.3 Звукові сигнали

Звукові сигнали є новим та цікавим способом обміну даними. Станом на 2018 рік, лише одна з відомих компаній «Chirp», займається розробкою та впровадженням цієї технології.

Технологія Chirp Connect дозволяє додаткам надсилати та отримувати інформацію за допомогою звуку. Chirp кодує масив байтів як аудіосигнал, який може бути переданий будь-яким пристроєм з динаміком і отриманий будь-яким пристроєм з мікрофоном і Chirp Connect SDK. Він розроблений таким чином, щоб бути надійним на відстані декількох метрів у шумному, повсякденному середовищі.

Оскільки передача відбувається цілком за допомогою аудіосигналів, не потрібно підключення до Інтернету або попереднього з'єднання, а будь-який пристрій у межах діапазону слуху може отримувати дані.

Сигнали Chirp Connect можуть бути згенеровані на пристрої з корисної навантаження динамічних даних або записуються як аудіофайл для подальшого відтворення, наприклад звуковий штрих-код [9].

Наразі ця технологіє є адаптивною для багатьох платформ та технологій такі як iOS, Android, Arm, JavaScript, Python, macOS, Windows, WebAssembly, Audio API [9].

Висновок по розділу

В результаті аналізу технологій обміну даними були зроблені наступні висновки:

1) Інтернет зараз є одним з найпоширеніших технологій обміну даними. Позитивною стороною є те що завдяки протоколам обміну даних ми можемо передавати дані які дуже великі за обсягом, на дуже великі відстані, знаходячись будь де, де є доступ до мережі.
   Але головним недоліком також можна вважати те що користувач для обміну даними повинен бути підключеним до мережі інтернет.
2) QR – код є сучасним інформаційним засобом. Позитивна сторона є те що він може ефективно використовуватися в будь-якій галузі, полегшуючи передання, або зберігання даних без доступу до мережі інтернет. Негативні сторони це - за відсутності програмних засобів та необхідних апаратних пристроїв зчитати код практично неможливо та й закодованими можуть бути не корисні дані, а наприклад, перехід на шкідливе посилання. Також об’єм даних для передачі є фіксованим, що не дає змоги передавати дані великих за об’ємом. Також ще одним нюансом є те, що чим більше даних

кодується для передачі, тим більша вірогідність того що при зчитуванні виникатимуть помилки при декодуванні даних.

3) У цьому розділі було проаналізовано технологію Chirp Connect. Позитивною стороною цієї технології є те що можна здійснювати обмін даними на відстань декілька метрів за допомогою звукових сигналів. Обсяг даних які передаються можуть бути будь-якої величини. Також позитивною стороною можна вважати те що використання цієї технології можливо на будь-якому пристрої де є динамік та мікрофон. Недоліками можна вважати те що чим більший обсяг даних, тим довше будуть програватися звуки, також для того щоб пристрої мали змогу обмінюватися даними ця технологія повинна бути присутня на обох пристроях.

РОДІЛ 4
АНАЛІЗ СТАНДАРТІВ ШИФРУВАННЯ ДАНИХ

Однією з основних цілей криптографії є захист інформації від несанкціонованого доступу та використання при передачі і зберіганні даних. Для реалізації даної можливості інформація проходить процедуру шифрування, в процесі якої відбувається перетворення відкритих даних в зашифровані з використанням параметрів, визначених для обраної системи шифрування. При процесі розшифрування, відбувається зворотне перетворення зашифрованих даних у відкриті. Як в прямому, так і в зворотному процесі використовуються ключі.

На даний час, існуючі алгоритми шифрування можна розділити на два класи:

1. симетричні – створені для шифрування і дешифрування. Використовується один і той же ключ;
2. асиметричні – створені для шифрування і дешифрування. Використовується два ключі, відкритий і секретний, відповідно.

4.1 Симетричні алгоритми

Для реалізації симетричних алгоритмів характерні наступні основні ознаки:

- для шифрування і розшифрування застосовується один і той ж алгоритм;
- для шифрування і розшифрування застосовується один і той ж секретний ключ.

Симетричні алгоритми поділяються на:

1. потокові, в яких відбувається шифрування потоку даних. Такий підхід необхідний в тих випадках, коли інформацію неможливо розбити на блоки - скажімо, якийсь потік даних, кожен символ яких повинен бути зашифрований і відправлений куди-небудь, не чекаючи інших даних, достатніх для формування блоку. Тому алгоритми поточного

шифрування шифрують дані побітово або посимвольний. Деякі класифікації не поділяють блочне і потокове шифрування, вважаючи, що потокове шифрування – це шифрування блоків одиничної довжини.

2. блокові - дані шифруються по блоках. Інформація розбивається на блоки фіксованої довжини (наприклад, 64 або 128 біт), після чого ці блоки по черзі шифруються. Причому, в різних алгоритмах шифрування або навіть в різних режимах роботи одного і того ж алгоритму блоки можуть шифруватися незалежно один від одного або "зі зчепленням" - коли результат шифрування поточного блоку даних залежить від значення попереднього блоку або від результату шифрування попереднього блоку.
    1) Шифри перестановки (permutation, P-блоки) – перестановка елементів тексту, які залишаються незмінними;
    2) Шифри заміни (підстановки, substitution, S-блоки) - окремі символи вихідної інформації замінюються на інші зберігаючи при цьому своє становище в потоці інформації:
        - моноалфавитной (код Цезаря),
        - поліалфавітних (шифр Вижинера, циліндр Джефферсона, диск Уетстоун, Enigma).
    3) Складові:
        - DES (Data Encryption Standard, США);
        - AES (Advanced Encryption Standard, Rijndael);
        - FEAL-1, а потім FEAL-4, FEAL-8, (Fast Enciphering Algoritm, Японія);
        - IDEA / IPES (International Data Encryption Algorithm);
        - Improved Proposed Encryption Standard (фірма Ascom-Tech AG, Швейцарія);
        - ГОСТ 28147-89 (СССР); та інші.

Також блочне шифрування можливо:

1. Без зворотного зв'язку (ОС). Кілька бітів (блок) оригінального тексту шифруються одночасно, і кожен біт оригінального тексту впливає на кожен біт шифротексту. Однак взаємовпливу блоків немає, тобто два однакових блоку вихідного тексту будуть представлені однаковим шифротексту. Тому для шифрування випадкової послідовності бітів (наприклад, ключів) можна використовувати тільки подібні алгоритми. Прикладами є DES в режимі ECB (Electronic Codebook, електронний шифрблокнот) і ГОСТ 28147-89 в режимі простої заміни.
2. З зворотним зв'язком. Зазвичай ОС організовується так: попередній шифрований блок складається по модулю 2 з поточним блоком. В якості першого блоку в ланцюзі ОС використовується ініціалізуючі значення. Помилка в одному біту впливає на два інших блоки - помилковий і наступний за ним. Приклад - DES в режимі CBC (Cipher Block Chaining, ланцюжок блоків).

В даний час в системах захисту інформації використовуються складові алгоритми, які поєднують в собі як перестановку, так і заміну (Підстановку) блоків даних, що дозволяє зробити систему більш стійкою до злому.

Коротка інформація з історичною довідкою про деякі сучасних стандартах шифрування представлена в таблиці (4.1). дані про характеристиках: розмірність ключів, блоків інформації, кількість циклів зміни даних, деяких алгоритмів шифрування наведені в таблиці (4.2).

Таблиця 4.1

Коротка інформація про деякі стандарти шифрування

Стандарт шифрування	Коротка довідка час, місце розробки, автори
ГОСТ 28147-89	Стандарт Російської Федерації на шифрування і імітозащіти даних. Розроблено в другій половині 70-х років. Спочатку мав гриф секретності, потім гриф послідовно знижувався, і до моменту офіційного проведення алгоритму через Держстандарт СРСР в 1989 році був знятий. Алгоритм залишився «для службового користування». У 1989 році став офіційним стандартом СРСР, в подальшому, федеральним стандартом Російської Федерації. Також є стандартом в країнах СНД. Стандарт являє собою блоковий алгоритм з 256-бітовим ключем і 32 циклами перетворення, що оперує 64-бітними блоками. Перевагою алгоритму є безперспективність атаки методом перебору і високу швидкодію.
DES (Data Encryption Standard**)**	Розроблено в 1975 році в дослідницькій лабораторії корпорації IBM і прийнятий як стандарт Національним інститутом стандартизації США (ANSI) в 1981 році. Федеральний стандарт шифрування США в 1977-2001 роках Довжина криптографічного ключа - 56 біт. Розвиток обчислювальної техніки призвело до того, що повний перебір всіх 56-бітних ключів представляється можливим. Тому DES не може більше використовуватися в якості надійного засобу захисту електронної інформації. Його можна рекомендувати лише для цілей тестування. У грудні 2001 року втратив свій статус в зв'язку з введенням в дію нового стандарту.
AES (Advanced Encryption Standard) **(Rijndael)**.	Розроблено в 1997 році в Бельгії. В даний час є федеральним стандартом шифрування США. Симетричний алгоритм блочного шифрування (розмір блоку 128 біт, ключ 128/192/256 біт). Цей алгоритм добре проаналізований і зараз широко використовується. Наприклад, Secret Disk Crypto Extension Pack дозволяє використовувати алгоритм AES з довжиною ключів 128 і 256 біт.
Blowfish	Складна схема вироблення ключових елементів істотно ускладнює атаку на алгоритм методом перебору, проте робить його непридатним для використання в системах, де ключ часто змінюється, і на кожному ключі шифрується невеликі за обсягом дані. Алгоритм найкраще підходить для систем, в яких на одному і тому ж ключі шифруються великі масиви даних. 1993 год. Брюс Шнайер (Bruce Schneier)
Twofish	Був запропонований в 1998 році і розроблений на основі алгоритмів Blowfish, SAFER і Square. Інформація також шифрується 128-бітними блоками. Secret Disk Crypto Extension Pack дозволяє використовувати алгоритм Twofish з довжиною ключа 256 біт.

Таблиця 4.2

Технічні характеристики деяких складових алгоритмів шифрування

Назва алгоритму	Розмір ключу, біт	Розмір блоку, біт	Розмір вектору ініціалізації, біт	Кількість циклів шифрування
Стандарт DES 4 режими роботи: ECB (Electronic Codebook) електронний шіфрблокнот CBC (Cipher Block Chaining) ланцюжок блоків CFB (Cipher Feedback) зворотний зв'язок по шіфротекста (гамування) OFB (Output Feedback) зворотний зв'язок по виходу (гамування)	56 +8 проверка контроля четности	64	64	16
Стандарт ГОСТ 28147-89 4 режими роботи: простий заміни гамування гамування зі зворотним зв'язком вироблення имитовставки	256	64	64	32
AES	128, 192, 256	-	-	-
CAST (аналог DES) Відмикання шляхом прямого перебору	40-64	64	-	8
FEAL-1 (альтернатива DES)	64	64	4	4
Blowfish	32 – 448	64	-	16
B-Crypt	56	64	64	-
IDEA	128	64	-	8
Lucipher	128	128	-	-

4.1.1 Мережа Фейстеля

Блокові шифри, які використовують послідовність перестановок і підстановок, називають мережею перестановок-підстановок або SP-мережею. Більшість сучасних блокових алгоритмів, відносяться до так званим мереж Фейстеля. Це алгоритми або стандарти DES, Lucifer, FEAL, ГОСТ 28147-89, CAST, Blowfish і інші [19], [20].

На вхід блочного алгоритму шифрування подається блок відкритого тексту довжиною m-бітів і ключ K. Блок відкритого тексту поділяється на дві

частини - подблоки N1 і N2. Якщо розміри частин рівні, таку архітектуру називають класичною або збалансованої мережею Фейстеля. Якщо частини N1 і N2 не рівні, то алгоритм називають розбалансованою мережею Фейстеля.

Всі раунди обробки проходять по одній і тій же схемі. Спочатку виконується операція підстановки. Вона полягає в застосуванні до подблоков N2 деякої функції F використовує значення ключа, і подальшому додаванні отриманого результату з подблоков N1 за допомогою логічної операції XOR. Для всіх раундів функція раунду має одну і ту ж структуру, але залежить від параметра - підключа раунду. Підключи раундів відрізняються від загального ключа і один від одного і обчислюються на основі загального ключа K. Після підстановки виконується перестановка, що представляє собою обмін місцями двох половин блоку даних.

Результат i-го раунду визначається за результатом попереднього раунду:

$$\begin{cases} N1(i+1) = N2(i) \\ N2(i+1) = N1(i) \oplus F\big(N2(i), K(i)\big) \end{cases} \qquad (1\text{-}1)$$

де

i – раунд

N1, N2 - подблоки відкритого тексту

K (i) - ключ раунду I

Якщо параметри функції F змінюються після шифрування кожного наступного символу, то мережу Фейстеля є гетерогенною. Якщо параметри функції F не змінюються, то мережу - гомогенна.

Шифр, який використовує таку конструкцію звернемо. Процес розшифрування шифру Фейстеля принципово не відрізняється від процесу шифрування. Застосовується той же алгоритм, але на вхід подається шифрований текст, а підключи використовуються в зворотній послідовності.

Якщо розглядати криптоаналітичних стійкість алгоритму мережі Фейстеля, який є базою для симетричних алгоритмів шифрування, то можна виділити наступні основні параметри, від яких буде залежати стійкість алгоритму до злому:

- Розмір блоку. Чим більше розмір блоку, тим вище надійність шифру, але швидкість виконання операцій шифрування і розшифрування знижується.

- Розмір ключа. Чим довше ключ, тим вище надійність шифру, але при цьому швидкість виконання операцій шифрування і розшифрування знижується.

- Число раундів обробки. Чим більше число раундів шифрування, тим більше ускладнюється криптоаналіз шифру. У загальному випадку число раундів повинен вибиратися так, щоб всі відомі методи криптоаналізу вимагали більше зусиль, ніж аналіз за допомогою перебору всіх ключів.

- Алгоритм обчислення подключей. Чим складніше алгоритм обчислення подключей, тим більшою мірою ускладнюється криптоаналіз шифру.

- Функція раунду. Застосування гетерогенних і розбалансованих мереж Фейстеля може значно поліпшити характеристики шифру.

Основним недоліком симетричного шифрування є необхідність передачі ключів "з рук в руки". Цей недолік дуже серйозний, оскільки унеможливлює використання симетричного шифрування в системах з необмеженим числом учасників.

4.1.2 Стандарт DES

Блоковий алгоритмом з закритим ключем DES використовує мережу Фейстеля.

Наразі алгоритм DES широко використовувався при зберіганні і передачі даних між різними обчислювальними системами; в електронних системах, в поштових системах креслень, при електронному обміні комерційною інформацією.

DES володіє двома дуже важливими якостями блокових шифрів: лавини і повнотою:

- Лавинний ефект або розсіювання означає, що невеликі зміни в початковому тексті або ключі викликають значні зміни в зашифрованому тексті.
- Ефект повноти полягає в тому, що кожен біт зашифрованого тексту повинен залежати від багатьох бітів вихідного тексту.

При шифруванні алгоритмом DES повідомлення ділиться на блоки по 64 біта. Для кожного раунду відбувається генерація ключів з 64-бітного вихідного ключа, в якому значущими є тільки 56 бітів (7 байтів або 7 символів в ASCII). У процесі роботи алгоритму використовуються таблиці, за допомогою яких здійснюється шифрування інформації. Блок схема алгоритму DES представлена на малюнку 4.1.

Для кожного раунду алгоритму відбувається генерація ключів з початкового ключа. Під час перетворення відбувається циклічний зсув вліво на один або два біта в залежності від раунду, потім застосовується таблиця перестановок, після чого для остаточного раундового ключа відбираються 48 біт з 56 біт, тобто відбувається стиснення ключа.

До вихідного блоку інформації застосовується початкова перестановка, для здійснення якої використовується таблиця початкової перестановки (в реалізації des_IP_table). Після цього блок інформації розбивається на дві частини по 32 біта: ліва і права частини, до яких застосовується мережу Фейстеля протягом 16 раундів, при яких дані об'єднуються з ключем. На заключному етапі після 16 раунду права і ліва частини об'єднуються, виконується зворотна перестановка, для здійснення якої використовується таблиця зворотного перестановки (des_invIP_table).

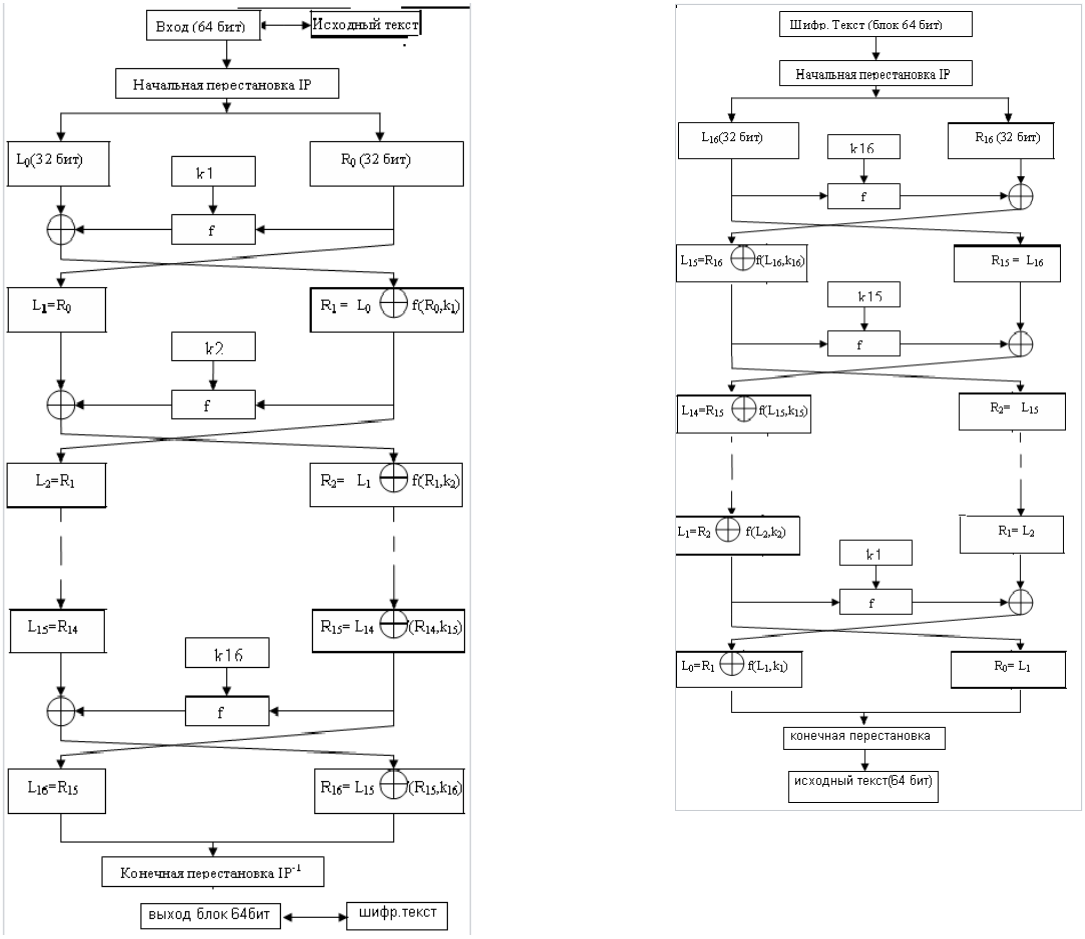


Рис. 4.1 - Блок-схема алгоритму DES. Шифрування і розшифрування

На рис. 4.2 і 4.3 представлена схема перетворень, що відбуваються в одному раунді алгоритму.

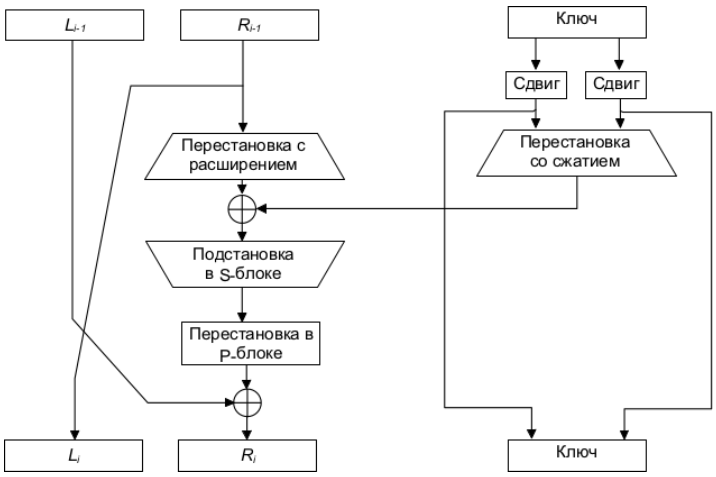


Рис. 4.2 - Раунд алгоритму DES

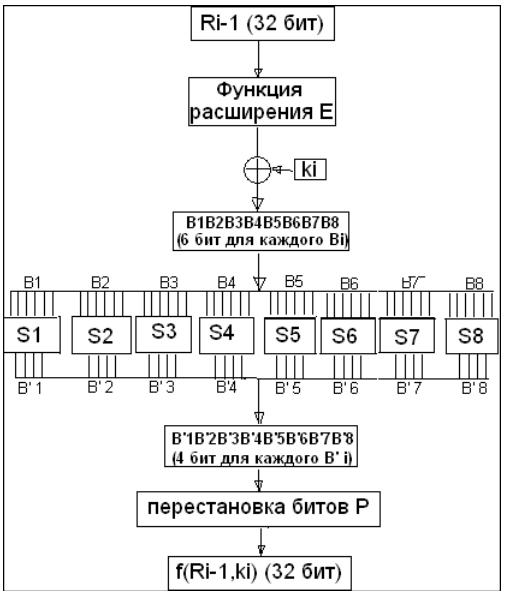


Рис. 4.3 - Представлення функції F алгоритму DES

Функція F описує дії в раунді являє собою послідовні перетворення, при яких дані об'єднуються з ключем (формула 1-1).

1. Застосування функції розширення E. Одна з двох частин інформації, наприклад, права, розміром 32 біт розширюється до 48 біт. Дублювання біт вихідної інформації здійснюється відповідно до таблиці розширення.

2. Операція XOR даних, отриманих на першому кроці, і ключа. Отриманий результат представляється у вигляді 8 блоків по 6 біт.

3. Застосування таблиці перетворення до отриманий блокам. Кожен з 8 блоків по 6 біт за допомогою таблиці перетворення трансформується в блок по 4 біта.

4. Застосування таблиці перестановки. До отриманих 32 бітам (8 блоків по 4 біта) застосовується таблиця перестановки і в результаті буде отримано перетворений блок.

При переході до наступного раунду права і ліва 32-бітові частини міняються місцями. Кроки з 1 по 4 повторюються.

Після закінчення всіх раундів перетворення, отримані 64 біта піддаються інверсії, для здійснення якої використовується таблиця зворотного перестановки (des_invIP_table) і, таким чином, буде отриманий зашифрований

текст.

Раундові ключі формуються з початково ключа. Додаються біти в позиції 8, 16, 24, 32, 40, 48, 56, 64 ключа k таким чином, щоб кожен байт містив непарне число одиниць. Далі до ключу застосовується таблиця перестановки, для кожного раунду застосовується циклічний зсув вліво, після чого із значущих 56 біт ключа вибирається 48 біт відповідно до таблиці.

Розшифрування відбувається в зворотному порядку з використанням того ж ключа.

З самого початку використання алгоритму криптоаналитики усього світу докладали багато зусиль для злому DES. Фактично DES дав різнобічний розвиток криптоанализа. Багато праці, присвячені різним методам криптоаналізу саме в додатку до алгоритму DES, а також деталей самого алгоритму і їх впливу на криптостойкость. Можна стверджувати, що в результаті проведених досліджень з'явилися цілі напрямки криптоанализа, такі як:

- Лінійний криптоаналіз
- диференціальне криптоаналіз
- аналіз залежностей між відкритим текстом і шіфротекста
- аналіз залежностей між співвідношеннями двох або більше відкритих текстів і відповідних їм шіфротекста
- кріптоаналіз на пов'язаних ключах
- пошук та аналіз залежностей між шіфротекста, отриманими на шуканому ключі і ключах, пов'язаних передбачуваним співвідношенням з шуканим ключем; проте DES виявився невразливий до даного виду атак, так як по ключовому розкладом циклічний зсув бітів ключа виконується на різну кількість позицій в різних раундах
- наявність слабких ключів.

Зусилля по злому DES робилися з 90-х років минулого століття:

Японський фахівець Міцуру Мацуї (Mitsuru Matsui), винахідник лінійного криптоаналізу, в 1993 році показав, що обчислити ключ шифру можна методом лінійного криптоаналізу при наявності у атакуючого 247 пар «Відкритий текст - шифротекст» [20].

Криптологи з Ізраїлю, винахідники диференціального криптоаналізу Елі Біхам (Eli Biham) і Аді Шамір (Adi Shamir) в 1991 році представили атаку, в якій ключ шифрування обчислювався методом диференціального криптоаналізу за умови, що атакуючий має 247 спеціально обраних пар «відкритий текст - шифротекст» [20].

Надалі ці атаки були кілька посилені (наприклад, атака лінійним Криптоаналіз при наявності 243 пар замість 247).

Як видно, всі атаки вимагають наявності величезної кількості пар «Відкритий текст - шифротекст» (таблиця 4.3), отримання яких на практиці є настільки трудомісткою операцією, що найбільш простий атакою на DES все ще можна вважати повний перебір.

Таблиця 4.3

Дані по способам криптоатак на алгоритм DES

Спосіб розтину	Результат	Передумова
Повний перебір	2 ^ 56	Потрібно 2 ^ 55 відомих пар текст-шифротекст
Лінійний криптоаналіз	2 ^ 50	Потрібно 2 ^ 43 відомих пар текст-шифротекст
Дифференціальний криптоаналіз	2 ^ 55	Потрібно 2 ^ 55 відомих пар текст-шифротекст

В даний час використовується Triple DES - потрійне шифрування, яке є більш стійким до злому. Для реалізації даного підходу потрібно два ключа: повідомлення шифрується першим ключем, розшифровується другим ключем і отриманий результат шифрується першим ключем. Також можлива реалізація з використанням трьох різних ключів. Всі подальші модифікації алгоритму спрямовані на те, щоб збільшити криптостойкость алгоритму. При цьому

алгоритм DES можна використовувати в якості тестового алгоритму при розробках методів криптоаналізу.

4.1.3 Стандарт AES

Ще одним симетричним алгоритмом, але з іншим типом архітектури є, так званий, «квадрат» або алгоритм Rijndael [23], який використаний в стандарті AES (Advanced Encryption Standard).

На відміну від мережі Фейстеля за раунд шифрування шифрується весь блок даних, а перетворення стосуються як окремих байтів масиву, так і рядків, і стовпців.

Відмінністю AES і Rijndael є розмір оброблюваного блоку і розміри ключів. - AES має фіксований розмір блоку 128 біт і розміри ключів 128, 192 і 256 біт. У разі Rijndael можуть бути задані будь-які розміри блоку і ключа, від 32 біт до 256 біт.

У загальному вигляді алгоритм AES представляє блок даних у вигляді двовимірного байтового масиву розміром 4X4, 4X6 або 4X8. Число раундів визначається: розмірністю ключа шифрування і розмірністю блоку даних. У таблиці 4.4 представлені можливі параметри алгоритму.

Таблиця 4.4

Параметри алгоритму AES

Ключ	Число 32-бітних слів ключа Nk	Число 32- бітних слів вхідних даних Nb	Число раунідв шифрування Nr
128 біт	4	4	10
192 біт	6	4	12
256 біт	8	4	14

Основним елементом алгоритму AES є байт - послідовність 8 біт, які обробляються, як єдине ціле. Для формування байтів 128-бітові блоку

відкритого тексту, вихідного блоку шифротексту і ключа шифру діляться на групи з 8-ми біт так, щоб в цілому вийшов масив байт.

Основні положення алгоритму:

- На першому етапі алгоритму відбувається формування стовпців матриці з масиву даних, які будуть шифруватися. Кожен елемент матриці є 8 біт:

in0	in4	in8	in12
in1	in5	in9	in13
in2	in6	in10	in14
in3	in7	in11	in15

- У результаті роботи алгоритму буде отримана матриця з елементами out:

out0	out4	out8	out12
out1	out5	out9	out13
out2	out6	out10	out14
out3	out7	out11	out15

- Проміжні результати раундів шифрування зберігаються в матриці state:

s 00	s 01	s 02	s 03
s 10	s 11	s 12	s 13
s 20	s 21	s 22	s 23
s 30	s 31	s 23	s 33

- Ключ для такого алгоритму також представляється у вигляді матриці, наприклад, для ключа довжиною 128 біт:

k 0	k 4	k 8	k 12
k 1	k 5	k 9	k 13
k 2	k 6	k 10	k 14
k 3	k 7	k 11	k 15

Чотири байта в кожному стовпці матриці станів і ключа можна розглядати

як одне 32-х бітове слово.

Тому матриця станів - це масив з 4 слів: w 0, w 1, w 2, w 3, де

w 0 { s 00 s 10 s 20 s 30 };

w 1 { s 01 s 11 s 21 s 31 };

w 2 { s 02 s 12 s 22 s 32 };

w 3 { s 03 s 13 s 23 s 33 }.

матриця ключа – це масив з 4 слів: w 0 , w 1 , w 2 , w 3 де

w 0 { k 0 k 4 k 8 k 12 };

w 1 { k 1 k 5 k 9 k 13 };

w 2 { k 2 k 6 k 10 k 14 };

w 3 { k 3 k 7 k 11 k 14 }.

Раундові ключі будуть формуватися з слів матриці ключа за допомогою спеціального алгоритму (див. Рис. 2.4).

Буде сформована послідовність з 44 слів (кожне слово по 32 біта): w 0, w 1, w 2, ..., w 43. Формування раундовий ключів здійснюється за допомогою розширення ключа. На кожен раунд шифрування подаються по чотири слова цієї послідовності.

Слова в ключовому масиві w 0, w 1, w 2, w 3 є 0-вим ключем.

Слова в ключовий матриці w 4, w 5, w 6, w 7 є 1-вим раундовим ключем Ключі для кожного наступного раунду формуються на підставі ключів попереднього:

$$w_{i+5} = w_{i+4} \oplus w_{i+1} w_{i+6} = w_{i+5} \oplus w_{i+2} w_{i+7} = w_{i+6} \oplus w_{i+3}$$

Перше слово в раундовий ключах формується таким чином:

$$w_{i+4} = w_i \oplus g(w_{I+3})$$

Функція g є послідовність операцій:

- Циклічний зсув вліво RotWord: {w 0, w 1, w2, w3} -> {w 1, w 2, w3, w0};
- - Заміна кожного отриманого байта SubWord здійснюється відповідно до Таблиці замін;
- Операція XOR даних з унікальною для кожного раунду постійної Rcon [i]

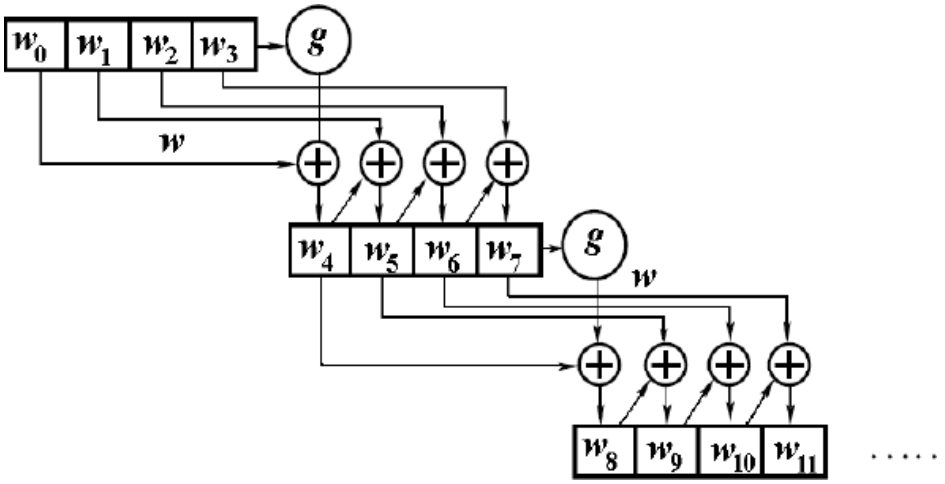


Рис. 4.4 - Формування раундових ключів

У кожному раунді алгоритму шифрування послідовно виконується чотири перетворення:

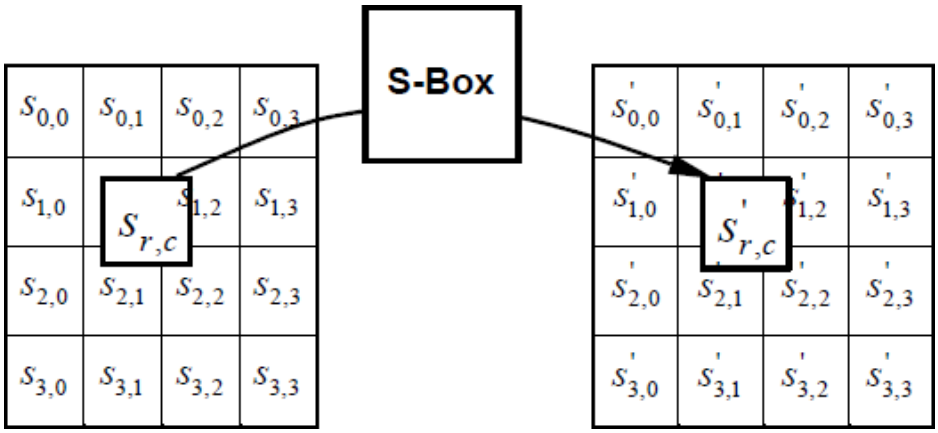
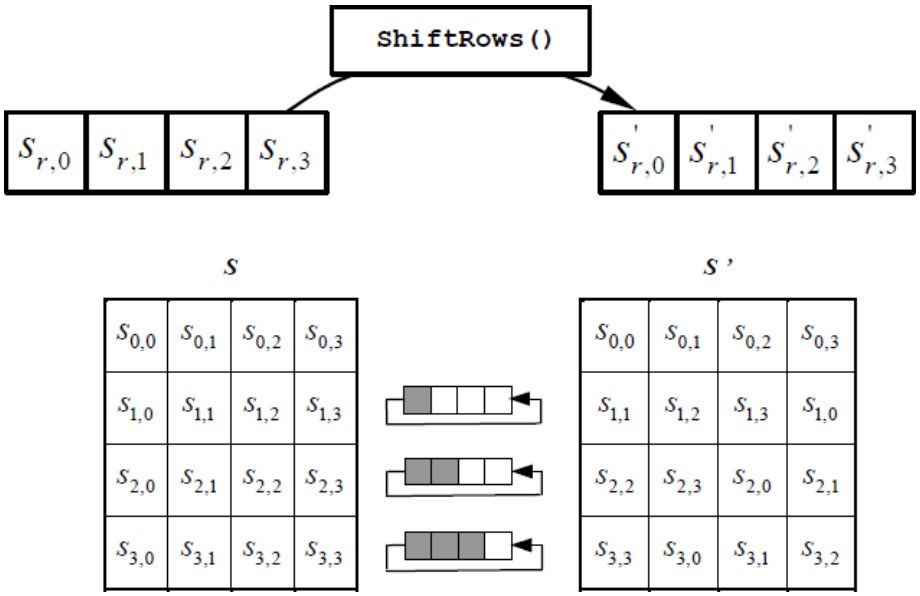


Рис. 4.5 - Операція табличної заміни

1. BS (ByteSub) - таблична заміна кожного байта масиву (див. Рис. 4.5) відповідно до таблиці замін Sbox;
2. SR (ShiftRow) - зсув рядків масиву (див. Рис. 4.6). При цій операції перший рядок залишається без змін, а інші циклічно побайтно зсуваються вліво на фіксоване число байт, залежне від розміру масиву. Наприклад, для масиву розміром 4X4 рядки 2, 3 і 4 зсуваються



відповідно на 1, 2 і 3 байти

Рис 4.6 - Операція зсуву рядків масиву

3. MC (MixColumn) - операція над незалежними стовпцями масиву (див. Рис. 4.7), коли кожен стовпець за певним правилом множиться на фіксовану матрицю [20]

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

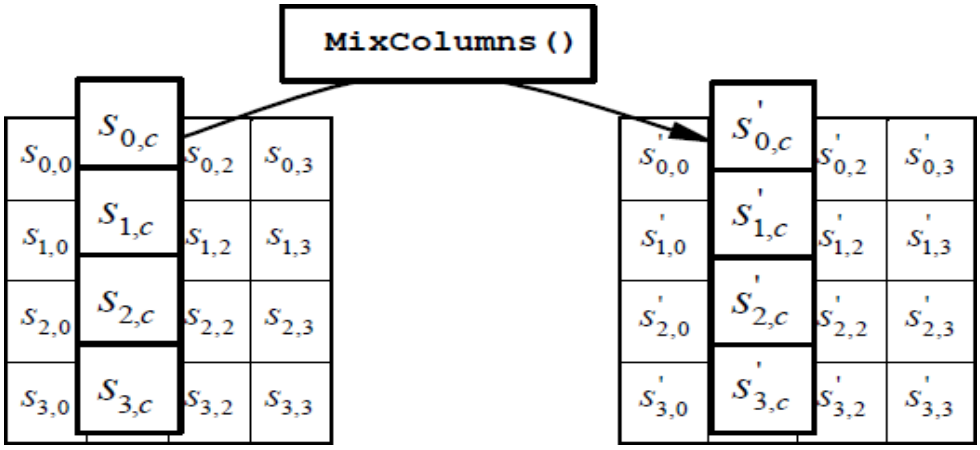


Рис. 4.7. Операція множення матриці на фіксовану матрицю c (x)

4. AK (AddRoundKey) - додавання раундового ключа. Кожен біт масиву складається по модулю 2 з відповідним бітом ключа раунду w (round * Nb + c) (див. Рис. 4.8).

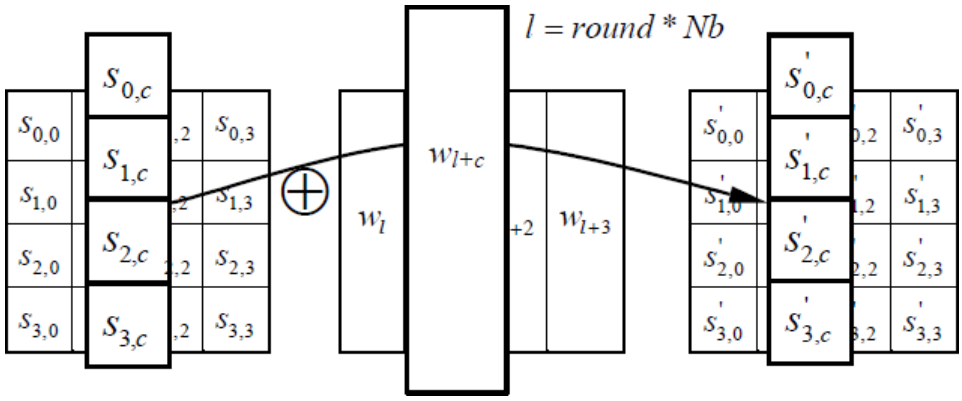


Рис. 4.8 - Операція додавання раундовий ключів

У кожному раунді (з деякими винятками) над шіфруемого даними по черзі виконуються перераховані перетворення (див. Рис. 4.9). Винятки стосуються першого і останнього раундів: перед першим раундом додатково виконується операція AK, а в останньому раунді відсутній MC. В результаті послідовність операцій при зашифрованими виглядає так: AK, {BS, SR, MC, AK} (повторюється R-1 раз), BS, SR, AK.

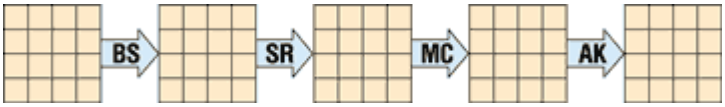


Рис 4.9 - Раунд алгоритму AES {BS, SR, MC, AK} (R-1 раз)

Раунд при розшифрування даних здійснюється за допомогою зворотних операцій:

1. Крок BS: Таблиця замін є инверсную таблицю, за допомогою, якої відбувалося шифрування;
2. Крок SR: Циклічне зрушення в операції SR відбувається вправо, а не вліво;
3. Крок MC: Множення в операції MC здійснюється на матрицю d (x), таку що d (x) * c (x) = 1
4. Крок AK: Повтор операції XOR з додаванням раундового ключа.

Стійкість алгоритму. У таблиці 4.6 показано кількість операцій необхідних при повному переборі.

Таблиця 4.6

Кількість операцій при атаці повним перебором для алгоритму AES

Ключ	Кількість операцій
128 біт	2 ^ 127
192 біт	2 ^ 191
256 біт	2 ^ 255

Розробники стверджують, що шифр стійкий проти:

- диференціального криптоаналізу;
- лінійного криптоаналізу;
- криптоаналізу на основі слабких ключів (слабких ключів в алгоритмі немає) З моменту публікації стандарту проводилися різні спроби розтину. Результати проведені в таблиці 4.7.

Таблиця 4.7

Результати спроб розтину алгоритму шифрування AES

Спроба розтину	Результат	передумова
Метод бумерангу	2 ^ 119	припущення, що відома пара відкритий текст - шифротекст
Метод біклік	2 ^ 126.1 2 ^ 254.1	Атака з підібраним шифротекста. Визначаються залежності шіфротекста від внутрішніх станів для фрагментів різних ключів. Gроісходіт перебір ключа по частинах
Атаки по побічним каналах:		Атаки не пов'язаний з математичними особливостями алгоритму.
Атака на основі часу виконання кожної операції	200 млн відібраних шифротекстів	-
Пошук випадкових апаратних помилок	2 ^ 32	-
Криптоаналіз по спожитої потужності	Успішно на чіпі ProASCI3	Перевірка дозволяє виявити апаратні зміни в мікросхемах, спрямовані на зняття криптографічного захисту, спотворення ключів.

4.2 Застосування алгоритмів шифрування

Шифрування використовується в різних сферах інформаційного захисту.

Поширена застосування: електронно-цифрові підписи, ідентифікація, шифрування інформації, що зберігається на носіях (жорстких дисках, знімних носіях), в мережах, хмарах, а також при передачі інформації між користувачами.

Сучасні алгоритми шифрування і розшифрування складні і їх неможливо виконувати вручну. У більшості додатків криптографія виробляється програмним забезпеченням і є безліч доступних криптографічних пакетів.

Наприклад, пакет Secret Disk - призначений для захисту конфіденційної інформації і персональних даних від несанкціонованого доступу, копіювання, ушкодження, крадіжки або примусового вилучення. За замовчуванням продукти Secret Disk можуть використовувати алгоритми шифрування DES і Triple DES, що надаються операційною системою.

Пакет розширення Secret Disk Crypto Extension Pack дозволяє використовувати алгоритм шифрування, що надається сторонніми

криптопровайдерами, такими як КріптоПро CSP, ViPNet CSP. Також Secret Disk Crypto Extension Pack містить додаткові алгоритми шифрування: AES, TwoFish.

Інформація, наведена нижче в таблиці 1.8, про деякі існуючих кріптопродуктах взята з джерела [24, 26].

Таблиця 4.8

Застосування криптосистем

Криптопродукт / Параметри	TrueCrypt	Secret	Zecurion Zdisk
Операційна система	Windows, Mac OS, Linux	Windows	Windows
Вбудовані алгоритми шифрування	AES, Serpent, Twofish	Hi	Hi
Використання постачальників криптографії (криптопровайдерів CSP)	Hi	Microsoft Enhanced CSP: Triple DES и RC2; Secret Disk NG Crypto Pack: AES и Twofish; КриптоПро CSP, Signal-COM CSP или Vipnet CSP: ГОСТ 28147-89	RC5, AES, КРИПТОН CSP: ДСТУ 28147-89
Каскадне шифрування	AES-Twofish-Serpent; Serpent-AES; Serpent-Twofish-AES; Twofish-Serpent	Hi	Hi
Режим шифрування XTS	так	Hi	Hi

Технологія шифрування XTS (XEX-based Tweaked CodeBook mode) - є розвитком попередніх блокових методів шифрування AES-XEX і LRW. Так як операції читання / запису на носіях інформації виробляються посекторного блоками, то використання потокових методів кодування неприйнятно. Метод шифрування XTS-AES для алгоритму AES був описаний і рекомендований міжнародним стандартом захисту інформації, що зберігається IEEE P1619.

Даний метод використовується, наприклад, в USB-накопичувачах компанії Kingston, в якому реалізований апаратним способом [20].

Алгоритм AES використовується для шифрування державних документів з грифами високого рівня секретності, також в таких областях, як захист банківських операцій, бездротовий зв'язок або шифрування даних на дисках.

В даний час AES широко використовується в продуктах, багато з яких сертифіковані в NIST [21]. Шифрування AES застосовується в архіваторах WinZip, WinRAR, 7zip. Також застосовується в ПО Folder Lock (шифрування файлів, вкладень електронної пошти або ін.) [25].

«Починаючи з версії 8.1.6 з Oracle поставляє пакет для шифрування і розшифровки методом DES під назвою DBMS_OBFUSCATION_TOOLKIT». У пакеті також міститься функція DES3GETKEY, що дозволяє згенерувати криптографічний коректний ключ. Пакет DBMS_CRYPTO, дозволяє шифрувати дані іншими алгоритмами не тільки DES, 3DES, але ще і AES, для цього використовуються дві функції ENCRYPT і DECRYPT.

Також в цьому пакеті є функція GETRANDOMBYTES, яка може використовуватися для формування криптографически випадкових ключів [22]. Алгоритм DES і 3DES і інші застосовуються для захисту фінансової інформації: так, модуль THALES (Racal) HSM 9000 повністю підтримує операції для емісії і обробки кредитних карт, EuroPay. Також цей модуль підтримує і інші криптографічні алгоритми [22].

Висновок по розділу

За результатами проведеного аналізу можна зробити висновок про те, що ефективність (відношення кількості коректно певних біт до довжини ключа) генетичного алгоритму краще, ніж мурашиного алгоритму.

Результати, отримані в ході проведеного аналізу показали, що AES є більш крипостійкість, ніж DES.

В цілому, використання евристичних алгоритмів криптоаналізу, підтверджують основне положення криптографічних алгоритмів: зміна одного біта в ключі призводить до лавинному ефекту.

РОЗДІЛ 5
АНАЛІЗ СУЧАСНИХ ХМАРНИХ СУБД

Існує постійно зростаюче число компаній, що надають хмарні послуги для бізнесу. Їх діапазон: від комунікацій і соціальних додатків до розгортання платформ і інфраструктур, які формують фундамент обчислювальних ресурсів компаній.

5.1 Веб-сервіс від компанії Amazon

5.1.1 Amazon Elastic Compute Cloud

Amazon Elastic Compute Cloud (Amazon EC2) - це веб-сервіс, який надає масштабуючу обчислювальну потужність в хмарі. Його головним завданням є полегшення розробки масштабованих веб додатків [41].

Простий і зрозумілий інтерфейс сервісу Amazon EC2 дозволяє отримати і налаштувати обчислювальну потужність з мінімальними зусиллями. Він надає користувачеві повний контроль над обчислювальними ресурсами і дає можливість працювати в надійній інфраструктурі Amazon. Сервіс знижує до декількох хвилин час необхідне для отримання і завантаження примірника сервера, дозволяючи швидко масштабувати обчислювальні ресурси як в більшу так і в меншу сторону в залежності від вимог користувача. В Amazon EC2 передбачена можливість оплати тільки за спожиті ресурси. Для розробників сервіс надає інструменти для побудови відмовостійких рішень, ізолюючи їх один від одного в аварійних ситуаціях.

Зашифрований файл образу сервера Amazon Machine Image (AMI), що знаходиться в хмарному сховищі Amazon S3, містить всю необхідну інформацію для завантаження примірника сервера і його програмного забезпечення [41]. Всі екземпляри, засновані на одних і тих же AMI образах, запускаються і працюють однаково. Будь-яка інформація на них втрачається в разі видалення або пошкодження образів.

Образ є інфраструктурної одиницею для розгортання робочого середовища. Є можливість мати тільки один такий образ або кілька, з яких може бути зібрана вся інфраструктура: веб-сервери, сервери додатків, сервери баз даних. Amazon EC2 надає набір командних інструментів для оперативного і простого створення образів AMI. Як тільки образ AMI був створений, його необхідно завантажити в сервіс сховища Amazon S3. Amazon EC2 використовує сервіс Amazon S3 для надання надійного, що масштабується сховища користувальницьких образів AMI.

Користувач сервісу може вибирати необхідний йому примірник сервера з бібліотеки образів. Наприклад, якщо користувачеві потрібен сервер на базі операційної системи Linux, то у нього є можливість вибрати спосіб на основі стандартних дистрибутивів Linux.

Amazon EC2 представляє справжню віртуальну обчислювальну середу, яка дозволяє використовувати інтерфейси веб-сервісу для запиту обчислювальних потужностей, завантаження їх за допомогою розробленого додатка, контроль над доступом, а також для запуску необхідної кількості серверів. Для використання сервісу Amazon EC2 досить [42]:

- Створити образ сервера Amazon Machine Image, що містить додатки, дані і настройки або використовувати вже існуючі налаштовані шаблони образів для негайного запуску серверів;
- Завантажити AMI в сервіс сховища Amazon S3, який надає безпечний, надійний і швидкий репозиторій для зберігання образів серверів;
- Налаштувати безпеку і мережевий доступ;
- Запустити, зупинити або спостерігати за активністю примірників образів за потребою за допомогою програмного інтерфейсу;
- Оплачувати тільки використовувані обчислювальні потужності.

5.1.2 Основні характеристики сервісу

Основними характеристиками сервісу Amazon EC2 є:

1. Масштабованість - Amazon EC2 дозволяє збільшувати або зменшувати обсяг використовуваних потужностей за кілька хвилин. Передбачена можливість одночасного запуску сотні або навіть тисячі серверів одночасно. Так як всі дії з серверами контролюються за допомогою програмного інтерфейсу, то користувальницький додаток, розгорнуте на сервері Amazon, може в автоматичному режимі масштабувати свої ресурси, в залежності від своїх потреб.

2. Повний контроль. Користувач сервісу має повний контроль над своїми екземплярами серверів. Йому надається найвищий рівень доступу до кожного сервера, який знаходиться в його розпорядженні. Примірники серверів можуть бути перезавантажувались віддалено через програмний інтерфейс сервісу.

3. Безпека. Amazon EC2 надає веб-сервіс для настройки брандмауера, який контролює мережевий доступ між групами серверів.

4. Практичність. Існує кілька типів примірників серверів, що дозволяє споживачеві вибрати необхідну конфігурацію пам'яті, процесора і сховища, яке оптимально для його робочого середовища.

5. Розроблено для використання з іншими сервісами Amazon. Веб-сервіс Amazon EC2 взаємодіє з сервісом зберігання Amazon Simple Storage Service (Amazon S3), сервісом баз даних Amazon SimpleDB і сервісом обробки повідомлень Amazon Simple Queue Service (Amazon SQS) для надання повного рішення для обчислень, обробки черг повідомлень і зберігання даних між безліччю різних додатків.

6. Надійність. Amazon EC2 пропонує отказоустойчивую середу, в якій можлива оперативна і безпечна заміна обладнання. Веб-сервіс працює в одному з найбільш надійних центрів обробки даних компанії Amazon [43].

Засоби для розробки відмовостійких додатків, які включають в себе [43]:

• Amazon EC2 надає можливість розміщувати екземпляри серверів в різних точках світу. Місця розташування складаються з регіонів і так званих «Зон доступу». Регіони географічно розділені й перебувають у різних

географічних областях чи країнах. «Гарячі точки це ізольовані зони, які були розроблені з метою запобігання зупинки сервісів в разі поломки однієї з таких зон і надають недорогу, з низькими затримками комп'ютерну мережу для доступу в інші« Гарячі точки »в одному регіоні. Регіони складаються їх однієї або декількох «Зон доступу». Запускаючи екземпляр сервера в різних «Зонах доступу», користувач захищає себе від аварії в одній з таких зон.

• Гнучкі IP адреси - це статичні IP адреси, створені для динамічних хмарних обчислень. Такі IP адреси асоціюються з призначеної для користувача обліковим записом і присвоюються конкретним екземплярів серверів.

5.1.3 Типи і вартість ресурсів, що поставляються сервісом

У розпорядженні сервісу Amazon Elastic Compute Cloud (Amazon EC2) знаходяться декілька різних типів примірників серверів для задоволення обчислювальних потреб користувачів сервісу. Кожен екземпляр являє собою заздалегідь налаштований обсяг потужностей, який може бути змінений в процесі експлуатації.

Примірники Amazon EC2 згруповані в дві різні категорії: стандартні і з великою кількістю обчислювальних одиниць. Стандартні екземпляри мають достатній обсяг оперативної пам'яті і обчислювальної потужності для більшості средненагруженних додатків. Примірники з великою кількістю обчислювальних одиниць мають порівняно менший обсяг оперативної пам'яті, ніж процесорних одиниць. Такі типи примірників найбільше підходять для високонавантажених додатків.

Кожен екземпляр має певну кількість зарезервованих ресурсів. Екземпляри цієї категорії підходять для більшості додатків Таблиця 5.1.

Таблиця 5.1

Стандартний екземпляр	Великий екземпляр	Екстра великий примірник
1.7 Гб пам'яті	7.5 Гб пам'яті	15 Гб пам'яті

1 обчислювальна одиниця	4 обчислювальні одиниці	8 обчислювальних одиниць
160 Гб сховища	850 Гб сховища	1,690 Гб сховища
32-х бітна платформа	64- х бітна платформа	64- х бітна платформа
$0.10 на годину	$0.40 на годину	$0.80 на годину

Для додатків, що вимагають велику кількість обчислювальних потужностей, існують такі екземпляри Таблиця 5.2.

Таблиця 5.2

Стандартний екземпляр	Великий екземпляр
1.7 Гб пам'яті	7 Гб пам'яті
5 обчислювальних одиниць	20 обчислювальних одиниць
350 Гб сховища	1690 Гб сховища
32- х бітна платформа	64- х бітна платформа
$0.20 на годину	$0.80 на годину

Перехід до моделі хмарних обчислень в корені міняє підхід розробників до придбання процесорних ресурсів. Замість покупки конкретного процесора і використання його протягом декількох місяців або років, розробник орендує необхідний набір ресурсів щогодини [44].

Так як Amazon EC2 побудований на стандартних апаратних засобах, з часом в розпорядженні веб-сервісу може з'явитися набір з різного фізичного обладнання, яке буде лежати в основі образів віртуальних серверів сервісу EC2.

Amazon EC2 використовує безліч способів для вимірювання об'єму спожитих ресурсів. Для того, що зробити процес порівняння ресурсів між різними типами примірників простим для розробників, Amazon ввів поняття обчислювальний одиниці: Amazon EC2 Compute Unit [44]. Кількість процесорів, яке знаходиться в розпорядженні конкретного екземпляра, виражається в обчислювальних одиницях Amazon EC2 Compute Unit. Одна така

обчислювальна одиниця еквівалентна процесору 2007 Opteron or 2007 Xeon з тактовою частотою 1.0-1.2 ГГц.

Крім вимірювання споживання обчислювальних ресурсів, Amazon EC2 веде облік обсягу переданих даних. Нижче представлена вартість за передачу даних:

- $ 0.100 за Гб - всі вхідні дані
- $ 0.170 за Гб - перші вихідні дані до 10 Тб в місяць
- $ 0.130 за Гб - такі вихідні дані до 40 Тб в місяць
- $ 0.110 за Гб - такі вихідні дані до 100 Тб в місяць
- $ 0.100 за Гб - вихідні дані більше 150 Тб в місяць

Використання гнучких IP адрес також є платною послугою і вартість її споживання така:

- $ 0.01 за невикористаний IP адреса в годину
- $ 0.00 за використаний IP адреса в годину до 100 серверів в місяць
- $ 0.10 за використаний IP адреса в годину більше 100 серверів в місяць

По закінченню місяця користувач буде повідомлений про необхідність оплати спожитих послуг. Кожен екземпляр буде працювати поки не траплятися наступні події: вимикання сервера користувачем, вимикання сервера за допомогою призначених для користувача додатків, видалення образу в зв'язку з поломкою програмного забезпечення або обладнання [45].

5.2 Хмарна операційна система від компанії Microsoft

Microsoft Windows Azure - це хмарна операційна система, яка служить для розробки і розміщення додатків, а також управління послугами середовища Azure Services Platform, яка в свою чергу надає розробникам обчислення за запитом, зберігання даних, масштабування, і управління додатками через мережу Інтернет за допомогою центрів обробки даних компанії Microsoft [46].

Продукт Windows Azure є відкритою платформою, яка підтримує мови програмування і середовища розробки як від компанії Microsoft, так і від інших виробників.

Для побудови додатків і служб на Windows Azure, розробники можуть використовувати свої існуючі рішення, спроектовані в середовищі Microsoft Visual Studio. Windows Azure не є системою, яка реалізує грід-обчислення, крім того Windows Azure можна назвати стандартним сервісом для подання послуг розміщення додатків та публікації їх в мережу Інтернет [47]. Важливо розуміти, що Windows Azure - це інтегрована, сервіс орієнтована і керована середовище розробки. Це середовище включає в себе можливості по надійному та ефективному обчисленню та зберігання даних, а також підтримку різного роду інструментів і протоколів з розробки.

Більшість існуючих додатків побудовані на наборі таких інструментів як: Linux, Apache, MySQL та PHP. У той час як компанія Microsoft рекомендує будувати додатки на її платформі .NET, використовуючи її ж інструменти розробки, очевидно, що обмежуючи Windows Azure платформою .NET і пропрієтарними інструментами, є ризик зниження відсотка використання середовища Windows Azure, що в свою чергу може привести до уповільнення її розвитку [48].

Відповідно до цими ризиками компанія Microsoft представляє пакет програмного забезпечення Windows Azure SDK for PHP, який дозволяє розробляти програми на основі технологій: Linux, Apache, MySQL та PHP. Даний пакет надає доступ до сховища, обчисленням та управління Windows Azure, за допомогою відкритого інтерфейсу REST / XML.

Будь-сервер доступний через мережу Інтернет може взаємодіяти з Windows Azure, навіть через додатки, написані на мові PHP.

На сьогоднішній день Windows Azure підтримує останню версію платформи .NET, а саме .NET Framework 4.0, що дозволяє використовувати більш просунуті інструменти, такі як перегляд статусу сервісів Windows Azure,

а також отримувати доступ тільки на читання до даних прямо з середовища розробки Visual Studio. Налагодження додатків Windows Azure стала набагато простіше завдяки інструменту IntelliTrace, який зберігає зневадження про стан програми. Розробка в Windows Azure може бути реалізована безпосередньо з середовищ розробки, а не через портал Windows Azure [49]. Сервіс баз даних Windows Azure підтримує просторові типи даних і бази даних об'ємом до 50 Гб. Також в Windows Azure є в наявності сервіс по синхронізації даних між множинами центрами обробки даних.

Перевага хмарного рішення від компанії Microsoft в тому, що вона пропонує користувачам ту ж саму функціональність, яку вони очікують, якби самі встановлювали програмне забезпечення [49]. Вирішальним моментом у виборі даного рішення є ціна використання послуг Windows. Основні цінові параметрами представлені у таблиці 5.3.

Таблиця 5.3

Вычисления	$0.12 / в час
Хранения	$0.15 / 1 Гб / в месяц
Хранение транзакций	$0.01 / 10 000 транзакций
Передача данных	$0.10 входящие / $0.15 исходящие / Гб

5.3 Платформа розробки від компанії Google

Firebase — це платформа для розробки застосунків та мобільних додатків. Firebase розвивається з 2011 року компанією Firebase Inc., яка була придбана Google у 2014[50].

5.3.1 Історія

Firebase зявився з Envolve, це попередній стартап, заснований Джеймсом Темпліном та Ендрю Лі в 2011 році. Envolve в свою чергу надавав розробникам API, який дозволяв інтегрувати на свої веб-сайти функціональні можливості

онлайн-чатів. Тамплін і Лі виявили після випуску сервісу чату, що мобільні застосунки використовують сервіс для передачі даних, які не були повідомленнями чату. Розробники мали ідею щоб використовувати Envolve для синхронізації даних застосунків, наприклад таких як стан гри в режимі реального часу серед своїх користувачів. Тамплін і Лі вирішили відокремити архітектуру реального часу та систему чату, яка працювала на ньому. Вони заснували окрему компанію Firebase в квітні 2012 року [50].

Уже в травні Firebase Inc. отримав початкове фінансування 2012 року. Також у червні 2013 року компанія знову збільшила фінансування [50]. Firebase у жовтні 2014 року була придбана компанією Google [50]. А вже у жовтні наступного року, компанія Google придбала Divshot, щоб об'єднати її з командою Firebase. З моменту придбання, компанія Firebase виросла всередині Google і розширила їхні послуги, щоб стати єдиною платформою для мобільних розробників. Наразі цей сервіс інтегрується з різними службами Google, для того щоб впроваджувати нові продукти та масштабувати сам проект в цілому. У січні 2017 року компанія Google придбала нові сервіси, такі як Fabric і Crashlytics з Twitter, щоб приєднати ці служби до команди розробників сервісу Firebase [50]. Firebase у жовтні 2017 року запустив Cloud Firestore, документ-орієнтовану базу даних, що є дуже зручним у використанні [50].

5.3.2 Служби і рішення для розробки

Firebase Analytics — безкоштовне рішення для оцінки застосунків, яке дає змогу ознайомитись із використанням застосунків та залученням користувачів [50].

Firebase Cloud Messaging (FCM) раніше відомий як Google Cloud Messaging (GCM) — це крос-платформна технологія для повідомлень і нотифікацій які приходять на додатки з платформою для Android, iOS, яка наразі є безкоштовною у використанні, що є вагомою стороною серед конкурентів [50].

Firebase Auth — це служба, для реєстрації та аутефікації користувачів, використовуючи лише код на стороні клієнта. Також ця служба підтримує соціальні логін-провайдери Twitter, Google, Facebook і GitHub а також Google Play Games. Ще вона включає в себе систему управління користувачами. Розробники за допомогою входу з електронної пошти та пароля можуть увімкнути автентифікацію користувача. Дані електронної пошти на паролю зберігаються в Firebase [50].

Realtime Database. Firebase надає в режимі реального часу бекенд та базу даних як службу. Ця служба надає розробникам готове API, який зберігає дані у хмарі Firebase та дозволяє синхронізувати ці дані застосунків між клієнтами [50]. Firebase клієнтські бібліотеки, які дозволяють інтеграцію із застосунками iOS, Android, Objective-C, Swift, JavaScript / Node.js, Java. БД також доступна через REST API та прив'язки до декількох сценаріїв JavaScript, таких як Ember.js, React, AngularJS та Backbone.js [50]. REST API використовує протокол подій із сервером які відбуваються в базі даних реального часу, який є інтерфейсом для створення HTTP-з'єднань для отримання push-повідомлень на девайс користувача від сервера. Розробники, які використовують Realtime Database, мають захист даних за допомогою правил безпеки, що застосовуються на сервері Firebase [50].

Cloud Fiestore є більш потужнішою частиною Firebase Realtime Database, була випущена у бета-версії. У ній також розробники можуть слухати події які відбуваються з БД, але новою особливістю є те, що розробники можуть відслідковувати транзакції запитів.

Firebase Storage забезпечує надійне сховище для файлів, де можна робити завантаження та вивантаження файлів для застосунків Firebase незалежно від якості інтернет з’єднання. Кожен розробник може використовувати його для зберігання відео, аудіо, зображень чи іншого вмісту, створеного користувачами мобільних додатків. Зберігання даних Firebase підтримується завдяки Google Cloud Storage [50].

Firebase Hosting — це веб-хостинг статичного та динамічного типу , який було запущено 13 травня 2014 року. Він підтримує хостинг статичних файлів, таких як JavaScript, HTML, CSS та інші файли, а також динамічну підтримку Node.js бекенд через Cloud Functions. Служба передає файли через мережу доставки контенту (CDN) за допомогою протоколу HTTPS та SSL шифрування. Також Firebase підтримує Fastly, CDN, для того щоб забезпечити підтримку Firebase Hosting CDN. Хостинг Firebase виріс із запитів клієнтів – стверджує компанія; розробники потребували місця для розміщення їхнього вмісту тому що вони використовували Firebase для своєї бази даних в режимі реального часу [50].

ML Kit — це мобільна система, яка була запущена в режимі бета-тестування 8 травня 2018 року під час Google I/O 2018, для машинного навчання [50]. ML Kit API містить дуже різні інструменти, наприклад розпізнавання облич, розпізнавання тексту, сканування баркодів, створення опису для зображень та розпізнавання наземних об'єктів. Наразі вона доступна для iOS та Android розробників. Також можливий імпорт власних моделей TensorFlow [50]. API можна використовати у пристрої або у хмарі.

Основні цінові параметрами представлені в Додатку А.

Висновок по розділу

За результатами проведеного аналізу можна зробити висновок про те, що Firebase є набагато зручнішим та дешевшим у використанні порівняно з іншими сервісами хмарних СУБД. Також позитивними сторонами є те що Firebase надає безкоштовний доступ для користування багатьма своїми сервісами, наприклад Realtime Database, та готове API для цього сервісу, що в свою чергу поліпшує розробку мобільного додатку, а також зменшує час розробки. Але якщо цього API буде не достатньо, завжди можна скористатися Firebase Cloud Functions. Завдяки багатьом сервісам, а також технологіям Firebase, сервіс дає змогу розробляти дуже функціональні мобільні додатки, в дуже короткий час.

РОЗДІЛ 6
РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

4.1 Архітектура клієнт-серверного мобільного додатку

Розроблений мобільний додаток представляє собою клієнт двозвенної клієнт-серверної архітектури (малюнок 6.1). Двухзвенной називається архітектура, яка розподіляє три основні компоненти між клієнтом і сервером.

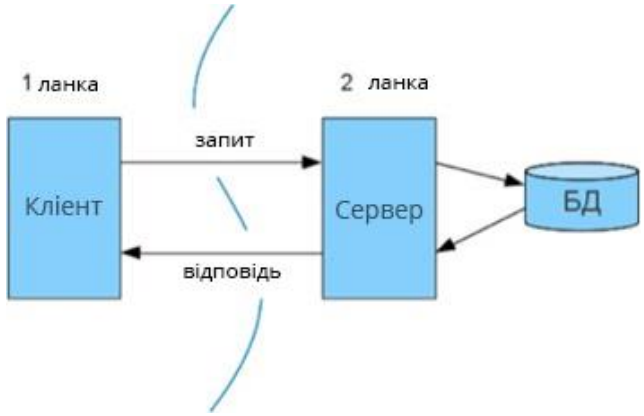


Рис. 6.1 - Архітектура програми

Клієнт - це програмне забезпечення надає користувачеві інтерфейс для взаємодії з системою. Клієнт не взаємодіє з базою даних безпосередньо, так само він не містить великої кількості бізнес-логіки. Клієнт відповідає за отримання і відображення даних отриманих з сервера. Так само клієнт відповідає за відправку отриманих від користувача даних, або виконує запит до даних по команді користувача.

У додатку для IOS, все взаємодії з користувачем відбуваються за допомогою сенсорного дисплея.

Архітектура IOS програми є 4 шари:

1. Cocoa Touch - містить в собі безліч фреймворків, які відповідають за зовнішній вигляд програми, а так само забезпечує базову

інфраструктуру додатки і підтримку основних технологій, таких як: багатозадачність, дотики, а push- повідомлення;

2. Media layer - містить в собі аудіо і відео технології, використовувані для реалізації мультимедійних функцій програми;
3. Core Services - складається з основних системних сервісів для додатків, де основними сервісами є Core Foundation і Foundation frameworks, що визначають основні типи використовуються усіма додатками, а так само він містить в собі технології для роботи з місцем розташування, iCloud, інтернет мережами;
4. Core OS layer - включає в себе такі сервіси як безпеку, локальна ідентифікація і Bluetooth.

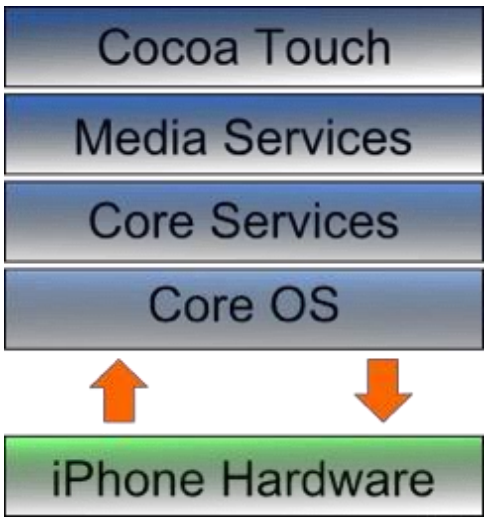


Рис 6.2 - Архітектура IOS програми.

Сервер - розташований на другому рівні логіки, містить велику частину бізнес-логіки системи. Сервер взаємодіє з базою даних, після цього генерує JSON моделі і відправляє їх клієнту через інтерфейс API. Так само відбувається процес обміну інформацією в іншу сторону, коли користувач змінює дані в додатку клієнта, потім вони відсилаються на сервер, після чого змінюється база даних.

Сервер Firebase взаємодіє з додатком для отримання такої інформації:

1. Під час реєстрування нового користувача:
    - інформацію про імені і прізвища користувача;
    - інформацію про себе від користувача;
    - інформацію про телефон користувача;
    - інформацію про эмейл користувача;
    - інформацію про місце проживання користувача;
    - інформацію про веб-сайт користувача;
    - головне фото користувача;
    - бекграунд фото користувача.
    - ідентифікатор користувача;
    - ідентифікатор пристрою користувача.
2. Під час пошуку нових користувачів через інтернет:
    - Інформацію про ідентифікатор відправника користувача;
    - Інформацію про місцезнаходження користувача;
    - Інформацію про радіус пошуку користувача.
3. Під час додавання нових користувачів за допомогую QR-коду:
    - Інформацію про ідентифікатор відправника користувача;
    - Інформацію про ідентифікатор отримувача користувача;
    - Інформацію про час створення QR-коду;
    - Інформацію про те чи валідний код;
    - Інформацію місцезнаходження під час зчитування коду;
    - Інформацію про час зчитування коду.
4. Під час додавання нових користувачів за допомогою звукових сигналів:
    - Інформацію про звуковий ідентифікатор відправника користувача;
    - Інформацію про звуковий ідентифікатор отримувача користувача;

- Інформацію місцезнаходження під час додавання користувача.
- Інформацію про час додавання користувача.

4.2 Розробка алгоритмів роботи програми

Була проведена розробка двох алгоритмів:

1) Алгоритм пошуку користувачів через інтернет. Даний алгоритм показує всю бізнес логіку пошуку користувачів через інтернет . У ньому представлено три аспекти алгоритму для Firebase, Backend а також самого мобільного додатку. Алгоритм представлений в Додатку Б.
2) Алгоритм додавання користувача за допомогую QR-коду. Розроблений алгоритм показує всю логіку роботи мобільного додатку для обміну даними через QR-код. З даним алгоритмом можна ознайомитися в додатку В.

Приклад генерації даних через QR-код.

```
- (UIImage *)createNonInterpolatedUIImageFromCIImage:(CIImage *)image
withScale:(CGFloat)scale
{
      // Render the CIImage into a CGImage
      CGImageRef cgImage = [[CIContext contextWithOptions:nil]
      createCGImage:image fromRect:image.extent];
      // Now we'll rescale using CoreGraphics
      UIGraphicsBeginImageContext(CGSizeMake(image.extent.size.width * scale,
image.extent.size.width * scale));
      CGContextRef context = UIGraphicsGetCurrentContext();
      // We don't want to interpolate (since we've got a pixel-correct image)
      CGContextSetInterpolationQuality(context, kCGInterpolationNone);
      CGContextDrawImage(context, CGContextGetClipBoundingBox(context),
      cgImage);
      // Get the image out
      UIImage *scaledImage = UIGraphicsGetImageFromCurrentImageContext();
      // Tidy up
```

```
        UIGraphicsEndImageContext();
        CGImageRelease(cgImage);
        return scaledImage;
}
```

Приклад отримання даних через QR-код.

```
- (void)captureOutput:(AVCaptureOutput *)captureOutput
didOutputMetadataObjects:(NSArray *)metadataObjects
fromConnection:(AVCaptureConnection *)connection
{
        if (metadataObjects != nil && [metadataObjects count] > 0) {
              AVMetadataMachineReadableCodeObject *metadataObj = [metadataObjects
              objectAtIndex:0];
             if ([[metadataObj type]
              isEqualToString:AVMetadataObjectTypeQRCode]){
                    dispatch_async(dispatch_get_main_queue(), ^{
                        [self stopReading];
                        [self createAlertWithParams:metadataObj.stringValue];
                    });
                }
            }
        }
}
```

Приклад отримання даних після обробки звукових сигналів.

```
[self.connect setReceivedBlock:^(NSData * _Nonnull data, NSUInteger channel)
    {
        if (data){
            NSLog(@"Received data: %@ on channel %lu", data, (unsigned long)
channel);
            NSString *charlieSendString = [[NSString alloc] initWithData:data
encoding:NSUTF8StringEncoding];
            [weakSelf.receivedLabel setText:charlieSendString];
        }
        else {
            NSLog(@"Decode failed.");
            [weakSelf.receivedLabel setText:@"Decode failed."];
        }
    }];
```

Приклад відправки даних для програвання звукових сигналів після кодування .

```
- (IBAction)sendButtonPressed:(id)sender {
    if (self.textView.text.length > 0) {
        NSData *payload = nil;
        if (_textView.text.length <= (self.connect.maxPayloadLength - 1))
            payload = [self.textView.text
dataUsingEncoding:NSUTF8StringEncoding];
        else {
            NSString *text = [self.textView.text
substringToIndex:(self.connect.maxPayloadLength - 1)];
            payload = [text dataUsingEncoding:NSUTF8StringEncoding];
        }
        NSError *error = [self.connect send:payload];
        if (error){
            NSLog(@"Error sending data: %@", error);
        }
    } else {
        NSUInteger length = 1 + arc4random() % (self.connect.maxPayloadLength -
1);
        NSData *payload = [[NSString stringWithFormat:@"%lu", length]
dataUsingEncoding:NSUTF8StringEncoding];
        NSError *error = [self.connect send:payload];
        if (error){
            NSLog(@"Error sending data: %@", error);
        }
    }
}
```

4.3 Розробка основних екранів додатку

Розробка програми проводилася в середовищі Xcode, за допомогою мови Objective C, по паттерну Apple MVC. Розмітка елементів екранів проводилася за допомогою Constranit, розмітка масштабується залежно від розміру екрана за допомогою «Auto-Layout».

Екран привітання дозволяє користувачу обрати наступний крок створити нового користувача, або увійти в старий аккаунт. Екран привітання містить три Lablel, дві Button, одну секцію ScrollView, один ImageView.

Рис. 6.3 – Екран привітання

Екран реєстрації дозволяє користувачу ввести логін і пароль та підтвердження паролю для створення акаунту в Firebase. Екран реєстрації містить два Label, три Button та три TextField.

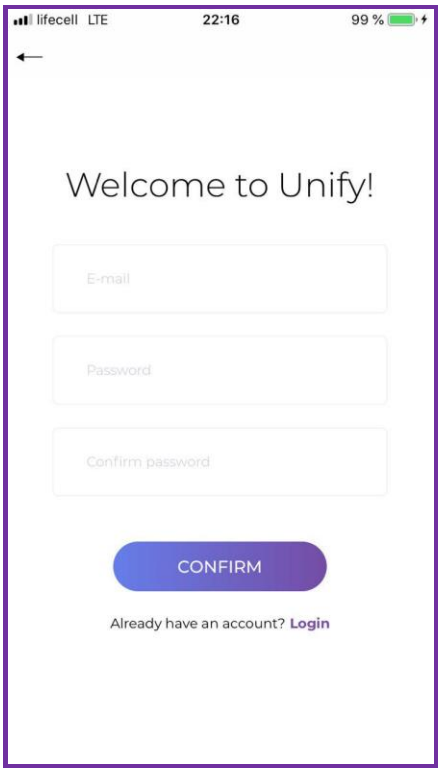


Рис. 6.4 – Екран реєстрації

Екран авторизації дозволяє користувачу ввести логін і пароль використовуваний в Firebase. Екран авторизації містить два Label, чотири Button і два TextField.

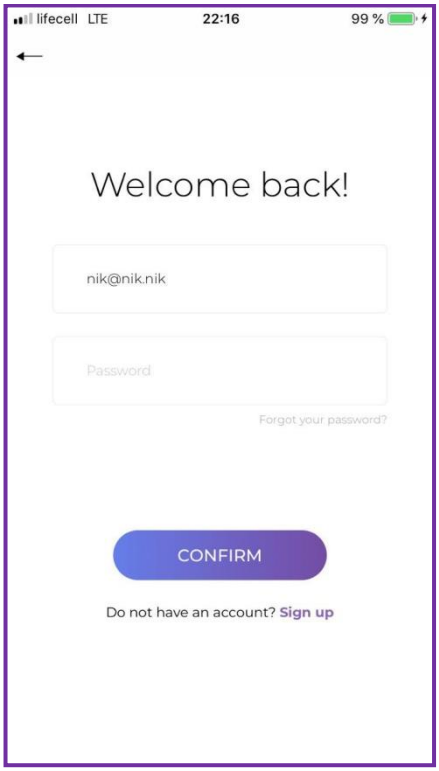


Рис. 6.5 – Екран авторизації

Екран відновлення паролю дозволяє користувачу ввести свій емейл який він використовує в Firebase, для того щоб відновити забутий пароль. Екран відновлення паролю містить один Label, дві Button і один TextField.

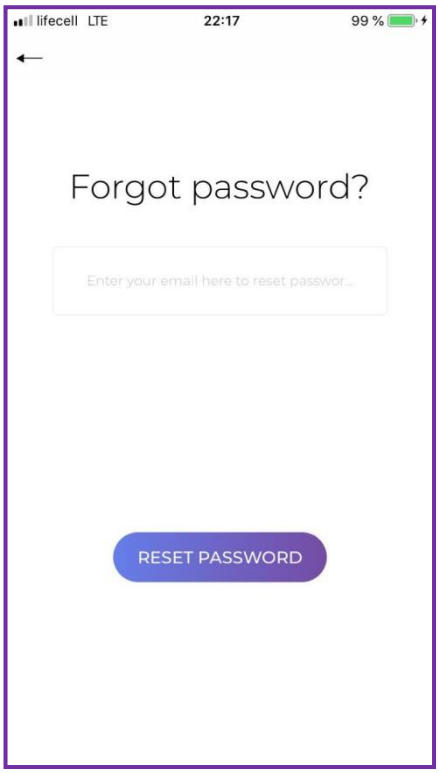


Рис. 6.6 – Екран відновлення паролю

Екран профілю дозволяє користувачу налаштувати свою сторінку з даними, та після натиску кнопки Done, користувач зберігає інформацію локально та на сервері. Екран профілю містить сім Label, сім Button, п’ять TextField, два ImageView, один ScrollView, один загальний TabBar.

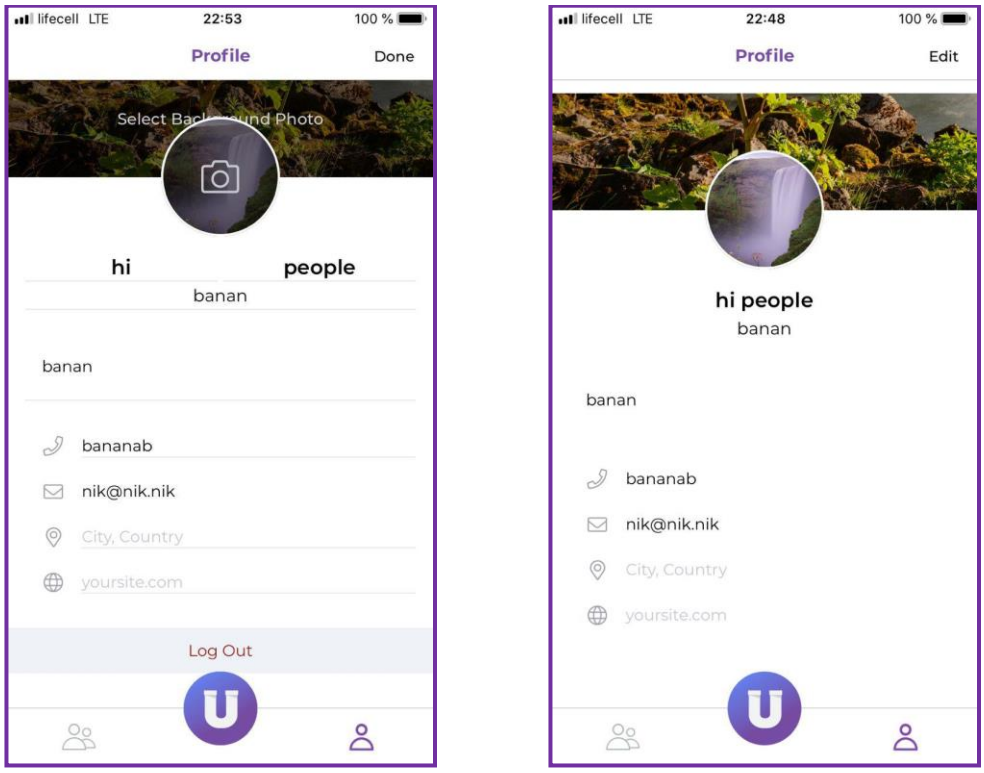


Рис. 6.7 – Екран профілю під час та після налаштування

Екран пошуку користувачів використовує три технології для обміну даними – це інтрернет, фреймворк «Chirp Connect» для використання звукових сигналів, а також QR-Код. Екран пошуку користувачів містить один Label, чотири Button, два ImageView, один загальний TabBar.

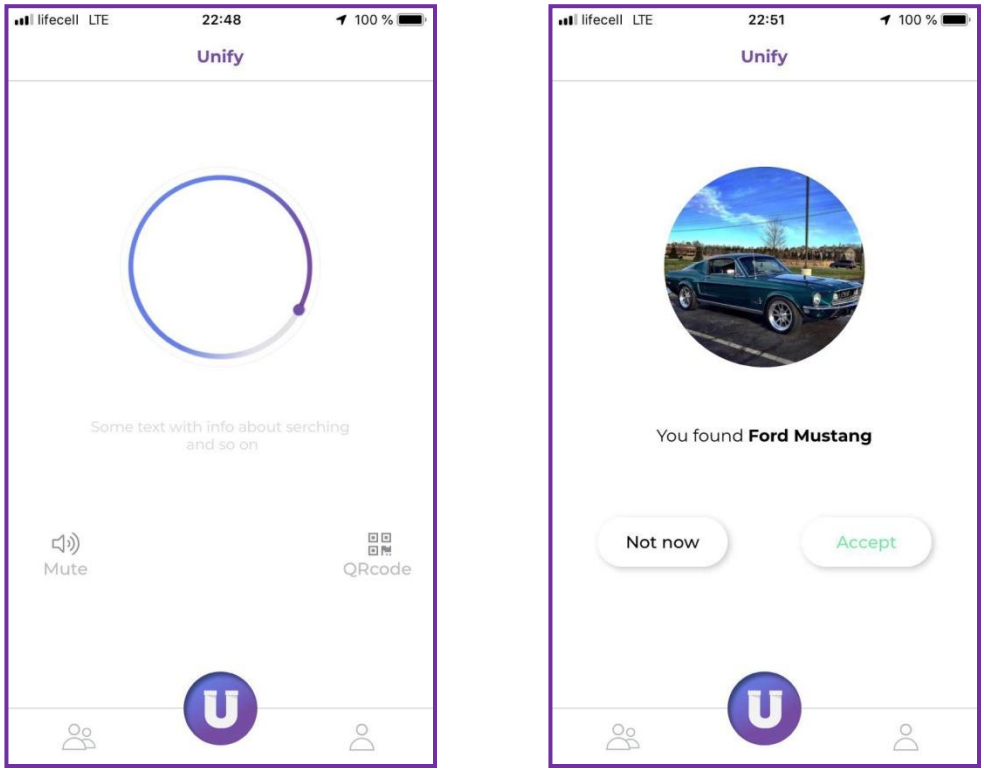


Рис. 6.8 – Екран пошуку користувачів під час пошуку та після знаходження за допомогою інтернету або звукових сигналів

Екран QR-коду одночасно генерує дані в QR-код, та починає зчитувати дані з камери. Екран QR-коду містить одну Button, один ImageView, один UIView, один AllertView для показу помилок або додавання користувача.

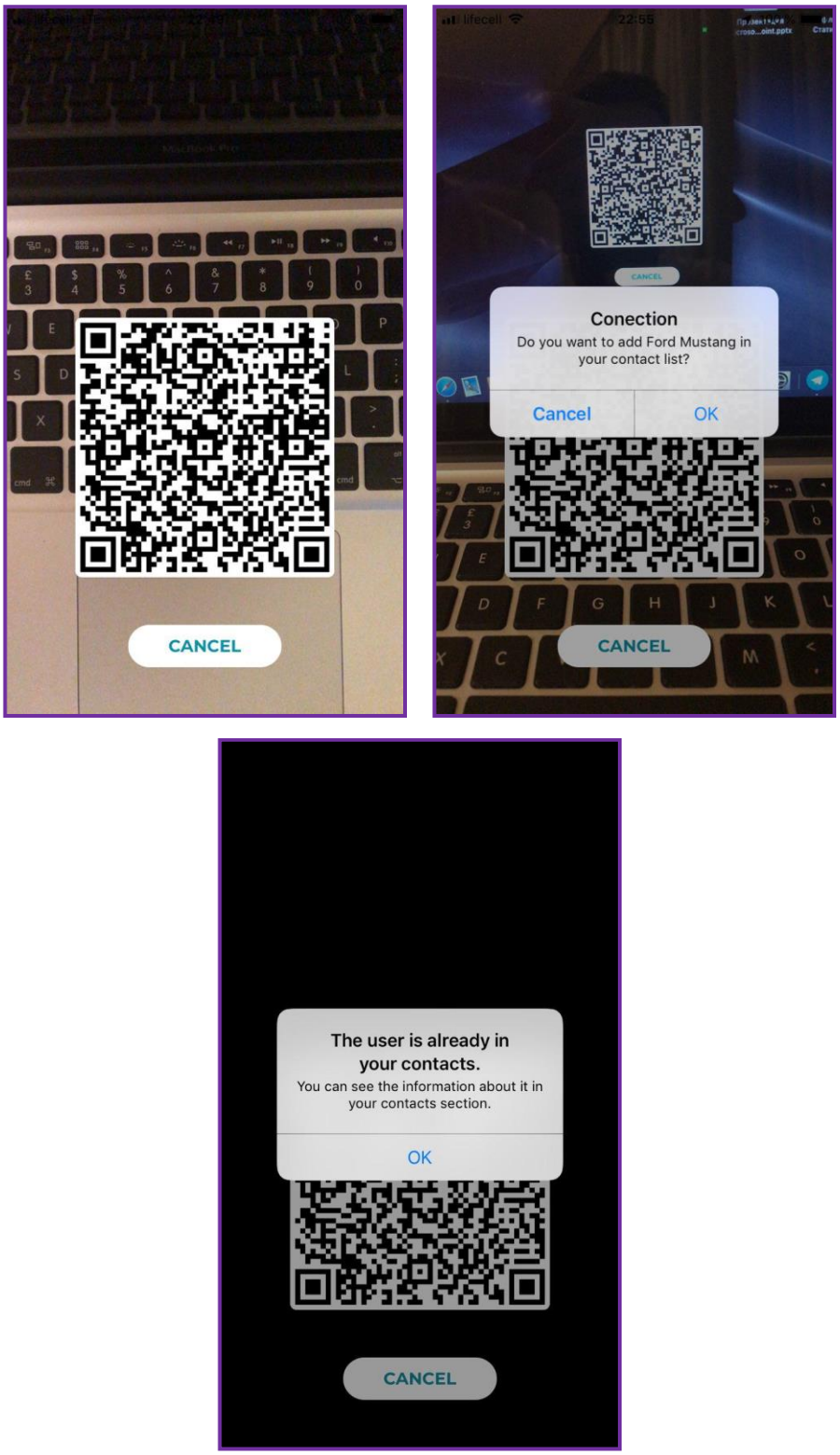


Рис. 6.9 – Екран QR-коду під час зчитування даних, додавання користувача та під час виникнення помилок

Екран списку контактів відображає контакті які відсортировані по алфавіту, або за датою додавання до списку. Екран пошуку користувачів містить один TableView, дві Button, один SearchBar, один загальний TabBar.

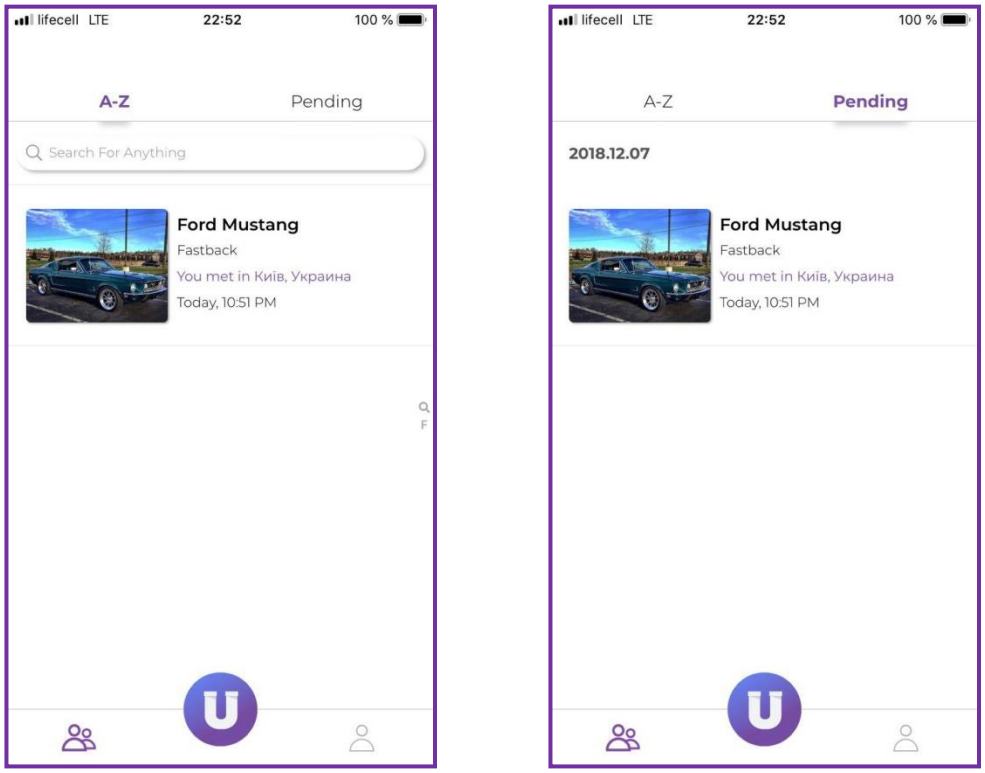


Рис. 6.10 – Екран списку контактів зліва – відсортований по алфавіту, заправа – за датою додавання до списку

Екран профілю доданого контакту. У ньому можливо переглянути інформацію про доданий контакт, видалити цей контакт, зробити нотатки, переглянути місце де користувачі додали один одного. де Екран профілю доданого контакту містить загальний TabBar, один ImageView, одинадцять Label, сім Button, один TextView, один MapView.

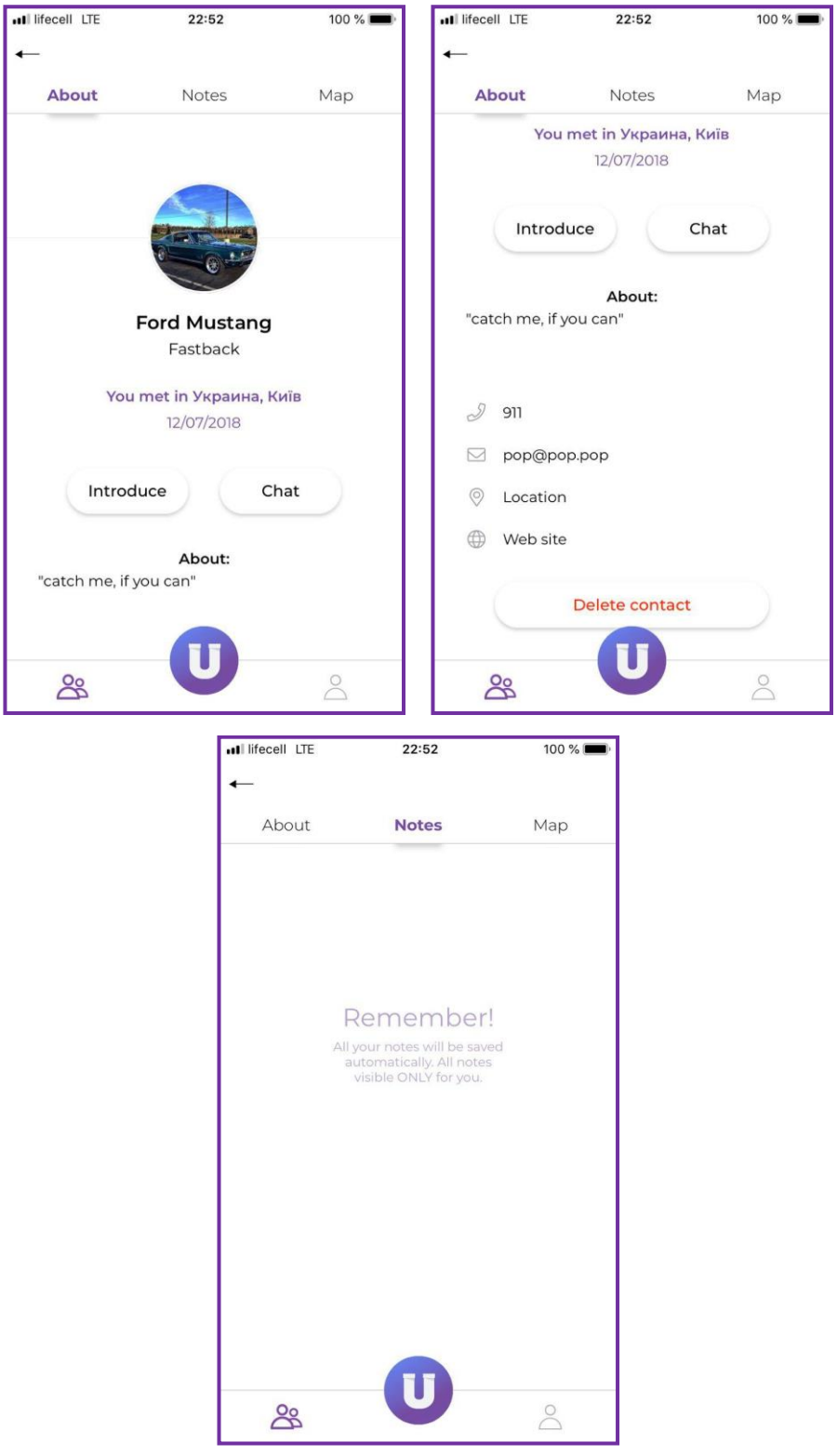


Рис. 6.11 – Екран профілю доданого контакту

Висновок по розділу

В цьому розділі було проаналізовано роботу клієнт-серверного мобільного додатку. Описано взаємодію сервісу Firebase з додатком, та дані які надходять для роботи з сервісом.

Було представлено два алгоритми роботи окремих частим мобільного додатку, загалом ці алгоритми описують обмін даними за допомогою інтернету та QR-коду.

Також представлені окремі частини коду, для обміну даними, враховуючи шифрування да розшифрування даними після зчитування QR-коду, а також методи роботи звукових сигналів.

Були вивчені та описані основні принципи дизайну інтерфейсу для мобільних пристроїв, які будуть використовуватися для подальшої функціональної розробки додатку.

Розроблено призначений для користувача інтерфейс програми, максимальне число переходів між екранами для виконання тестових сценаріїв дорівнює вісім, що дозволяє користувачу швидко здійснювати свою діяльність з додатком.

РОЗДІЛ 7
РОЗРОБКА СТАРТАП ПРОЕКТУ

7.1 Опис ідеї проекту

Розділ містить економічне обґрунтування стартап-проекту “Мобільний додаток обміну захищеними даними з використанням звукових сигналів”. Розділ має на меті ознайомлення з економічними та функціональними характеристиками майбутнього проекту, економічними аспектами його реалізації та впровадження у використання.

Опис основної ідеї майбутнього стартап-проекту наведено в Таблиці 7.1.

Таблиця 7.1

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Мобільний додаток обміну даними з використанням звукових сигналів	Підвищення швидкості обміну даними, покращення взаємодії між користувачами	Обмін даними в місцях без доступу до інтернету, полегшення взаємодії між користувачами

Мобільний додаток відрізняється від аналогів тим що має ще одну технологію для обміну даними – звукові сигнали.

П’ять основних факторів, що впливають на привабливість вибору ринку з огляду на характер конкуренції:

1. Основними конкурентами з мобільних додатків цього типу є Телеграм, Viber, WatsApp, Messenger.
2. Потенційними конкурентами є компанії, що розробляють мобільні додатки соціальних мереж.
3. Наразі використовуються мобільні додатки з технологіями обміну даними через Інтернет, та QR-код.

4. Постачальниками, що конкурують за ринку є Facebook, Viber, WatsApp, Telegram.
5. Споживачами є організації, бо що шукають нові шляхи обміну даними, або клієнти які часто подорожують.

Для оцінювання конкурентноспроможності та складності і можливості виходу стартапу на ринок було виконано порівняння з потенційними конкурентами, до яких можуть бути застосована обрана характеристика. Результати порівняння наведені в Таблиці 7.2.

Таблиця 7.2

Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні характе-риситики ідеї	(Потенційні) товари/концепції конкурентів				W слабка сторона	N нейтра-льна сторона	S сильна сторон а
		Мій проект	К-т 1	К-т 2	К-т 3			
1.	Форма виконання	Додаток	Додато к	Веб-сервіс	Додато к		+	
2.	Собівар-тість	Середня	Висока	Середня	Висока			+
3.	Наявність адміністра тора для налашту вання	Не треба	Не треба	Треба	Не треба		+	
4.	Наявність Інтернету	Не треба	Не треба	Треба	Не треба	+		
5.	Кросплатф ормність	Ні	Так	Ні	Так			+
6.	Масштабо ваність	Так	Ні	Ні	Ні			+

Проект декілька сильних сторін, що відсутні в аналогах та здатний конкурувати з ними. Сильними сторонами є відсутність необхідності в

адміністраторі для налаштування, низька собівартість реалізації, маштабованість, що дозволяє мати швидкий старт. Така перевага як масштабованість за у використанні нової технології обміну даними є відсутньою в потенційних конкурентів, тому є основною перевагою проекту і дозволяє розширяти його новими сервісами.

7.2 Технологічний аудит ідеї проекту

Проводиться аудит технологій, за допомогою якої може бути реалізована ідея проекту, тобто технології створення товару.

Таблиця 7.3

Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Мобільний додаток обміну даними з використанням звукових сигналів	Chirp	Наявна	Безкоштовна, доступна
		Objective C	Наявна	Безкоштовна, доступна
		Firebase	Наявна	Частково платна, доступна
Для створення сервісу обрані технології (Chirp, Obkcetive C, Firebase), які є безкоштовними, доступними та добре дослідженими потенційними розробниками.				

У таблиці Таблиця 7.3 надано результати огляду основних видів технологій, які можуть бути використані з метою реалізації мобільного додатку стартап-проекту. Було обрано декілька технологій, що не потребує доопрацювання та є безкоштовними.

7.3. Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які визначаються під час ринкового впровадження проекту, ринкових загроз, які також можуть перешкоджати реалізації проекту, дозволяє спланувати основні напрями розвитку проекту із урахуванням стану ринкового середовища, і також потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Було проведено аналіз попиту: обсяг, динаміка розвитку ринку, наявність попиту. Результати аналізу представлені у Таблиці 7.4.

Таблиця 7.4

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	8
2	Загальний обсяг продаж, грн/ум.од	15000 грн./ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Конфіденційність оброблюваних даних
6	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Дії, необхідні для виходу на такий ринок, залежать в тому числі і від потенційних споживачів. Аналіз цільових аудиторій споживачів даного продукту наведено у Таблиці 7.5.

Таблиця 7.5

Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Додатки, що спрощують процес обміну даними	Компанії які розробляють мобільні додатки	Очікується низька зацікавленість в стартапі великих компаній	Надійність, захищенність та конфіденційність процесу

Відповідно до результатів аналізу цільових аудиторій споживачів описаного продукту, наведеного у Таблиці 7.5 та необхідного для виходу на такий ринок, який було описано в Таблиці 7.4, слід спрямовувати зусилля на активне просування проекту в приватних клініках.

Важливим процесом для виходу на ринок та орієнтації на певну цільову аудиторію споживачів є аналіз можливих загроз стартап-проекту, що можуть спричинити значні проблеми для його розвитку. Результати відповідного аналізу факторів загроз продукту наведено в Таблиці 7.6.

Таблиця 7.6

Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Динаміка ринку	Уповільнення росту ринку	Співпраця з іншими компаніями для поліпшення ситуації на ринку
			Розширення на суміжні ринки
2.	Конкуренція	Вихід на ринок	Вихід з ринку
			Запропонувати великій компанії поглинути себе

		великої компанії	Надати додаткові переваги власного продукту лише за появи сильного конкурента
3.	Конкуренція	Поява іноземних конкурентів з товарами низької вартості	Акцентування уваги на якості результату, що надає продукт (або ж на будь якій інший явній перевазі відповідно до Таблиці 4.2)
4.	Потреби користувачів	Користувачам необхідний сервіс з іншим функціоналом	Надання нового функціоналу вже існуючій системі (завдяки її адаптивності)
5.	Держава	Зростання податкового тягаря	Перегляд виконання умов, що зменшують податки
			Поступове підвищення тарифів

Відповідно до результатів аналізу можливих факторів загроз стартап-проекту описаного продукту, наведеного у Таблиці 7.6 та необхідного для виходу на ринок, описаний в Таблиці 7.4, існує ряд ризиків які слід враховувати при планування виходу продукту на ринок та мати орієнтовні сценарії їх мінімізації та компенсування їх впливу, наведені в таблиці вище.

Аналогічно до загроз стартап-проекту, що можуть спричинити значні проблеми для його розвитку, важливою частиною також є огляд можливих сприятливих умов. При використанні яких можна покращити становище спартап-проекту та надати йому певну перевагу порівняно із конкурентами. Такі сприятливі умови та відповідні можливості розглянуто в Таблиці 7.7.

Таблиця 7.7

Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Можливості користувачів	Зростання можливостей потенційних покупців	Таргетингова реклама
2.	Конкуренція	Зниження довіри до конкурента 1 внаслідок помилок в функціонуванні	Акцентувати увагу на надійності системи

3.	Конкуренція	Зменшення числа конкурентів за рахунок появи бар'єрів входу на ринок	Заохочення співробітників конкурентів до зміни компанії
4.	Технології	Поява нових технологій	Аналіз технології, включення нових модулів (що технологію використовують)
5.	Держава	Послаблення обмежень в законодавстві	Оптимізація діяльності для скорочення витрат

Відповідно до результатів аналізу можливих сприятливих умов для описаного продукту, наведеного у Таблиці 7.7 та вкрай бажаного для виходу на ринок, описаний в Таблиці 7.4, існує ряд можливих реакцій компанії, правильне застосування яких може надати значну перевагу порівняно із конкурентами.

Конкуренція на ринку може стати як причиною занепаду компанії, так і стимулом завдяки якому стартап-проект значно покращить якість послуг та отримає корисний для майбутнього розвитку досвід. У зв'язку з цим в Таблиці 7.8 проводиться аналіз пропозиції: визначаються напрями конкуренції на ринку стартап-проекту.

Відповідно до результатів аналізу пропозиції описаного продукту, наведеного у Таблиці 7.8 та необхідного для виходу на ринок, описаний в Таблиці 7.4, визначено загальні риси конкуренції на ринку цього проету, та зазначено можливий вплив на діяльність підприємства страттап-проекту – тому можливі дії та заходи компанії, спрямовані на підвищення її конкурентоспроможності.

Таблиця 7.8

Ступеневий аналіз конкуренції на ринку

Характеристика конкурентного середовища	Особливість конкурентного середовища	В чому проявляється характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
Тип	Досконале	Існує 3 фірми-конкуренти на ринку	Врахувати ціни конкурентних компаній на початкових етапах створення бізнесу, реклама (вказати на конкретні переваги перед конкурентами)
Рівень конкурентної боротьби	Міжнародне	Один зарубіжній конкурент	Додати можливість вибору мови ПЗ, щоб легше було у майбутньому вийти на міжнародний ринок
Галузева ознака	Внутрішньогалузеве	Конкуренти мають ПЗ, який використовується лише всередині даної галузі	Створити основу ПЗ таким чином, щоб можна було легко його переробити для використання у інших галузях
Вид товарів	Товарно-видове	Однаковий вид товарів (ПЗ) і послуг (сфера медицини)	Створити ПЗ, враховуючи недоліки конкурентів
Характер конкурентних переваг	Нецінове	Вдосконалення технології створення ПЗ, для зменшення його собівартості	Використання менш дорогих технологій, більш ефективних методологій
Інтенсивність	Немарочне	Бренди відсутні	-

Як було зазначено вище, конкуренція відіграє надзвичайно важливу роль у розвитку компанії. У зв’язку з цим в Таблиці 7.9 наведено аналіз конкуренції в галузі за кількома її складовими, для кожної з якої наведено висновки.

Таблиця 7.9

Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти (бар'єри входження в ринок)	Фактори сили постачальників	Фактори сили споживачів	Фактори загроз з боку товарів-замінників
Висновки	Існує 3 конкуренти на ринку Найбільш схожим за виконанням є конкурент 1	Можливості для входу на ринок наявні Надане рішення спрощує та пришвидшує роботу спеціаліста.	Постачальники відсутні	Важливим для користувача є зручність у користуванні	Використання дешевних технологій створення ПЗ Менша собівартість товару

Відповідно до результатів аналізу пропозиції описаного продукту, наведеного у Таблиці 7.9 та необхідного для виходу на ринок, описаний в Таблиці 7.4, визначено основні особливості конкуренції на ринку стартап-проекту та зазначено висновки щодо кожної зі складових проведеного аналізу для успішної конкуренції на ринку.

На основі аналізу конкуренції із урахуванням основних характеристик ідеї проекту, вимог споживачів до товару та факторів маркетингового середовища (Таблиці 7.8 та 7.9) визначається та обґрунтовується перелік факторів конкурентоспроможності що наведено у Таблиці 7.10.

Таблиця 7.10

Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентноспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Кросплатформність	Відсутність жорстких вимог до платформи для встановлення
2.	Масштабованість та адаптивність	Без додаткових зусиль розширюється як функціональність проекту, так і апаратне забезпечення, необхідне для роботи

Відповідно до результатів аналізу факторів конкурентоспроможності продукту, наведеного у Таблиці 7.10 та необхідного для виходу на ринок, описаний в Таблиці 7.4, обґрунтовано головні фактори конкурентоспроможності, такі як масштабованість, кросплатформність та адаптивність, переваги яких також було розкрито в Таблицях 7.6 та 7.7.

На основі результатів аналізу факторів які зазначені вище конкурентоспроможності даного стартап-проекту було проведено аналіз сильних та слабких його сторін, а результат якого надано в Таблиці 7.11 за використання оцінки конкурентноспроможності за 20-бальною шкалою.

Таблиця 7.11

Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентноспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з нашим підприємством						
			-3	-2	-1	0	+1	+2	+3
1.	Новизна	20		+					
2.	Масштабованість та адаптивність	20	+						
3.	Необхідність в доступі до Інтернету	3							+

Відповідно до результатів аналізу сильних та слабких сторін проекту, наведеного у Таблиці 7.10 та необхідного для виходу на ринок, описаний в Таблиці 7.4, визначено, що найбільшою перевагою стартап-проекту є його адаптивність та маштабованість, що робить можливим надання великої частини нового функціоналу вже існуючій системі;

Результати аналізу, що надані в Таблиці 7.11, використано для проведення SWOT-аналізу, що є фінальним етапом ринкового аналізу можливостей впровадження проекту. Результат SWOT-аналізу представлено у вигляді таблиці-матриці аналізу слабких-Weak та сильних-Strength сторін стартап-проекту, зазначених в Таблиці 7.2, загроз-Troubles з Таблиці 7.6 та можливостей-Opportunities з Таблиці 7.7.

Таблиця 7.12

SWOT-аналіз стартап-проекту

S	Можливість надання нового функціоналу вже існуючій системі, відсутність жорстких вимог до платформи	Необхідна кроссплатформлена розробка	W
O	Таргетингова реклама, акцентуваня уваги на надійності системи, заохочення співробітників конкурентів до зміни компанії	Надання додаткових переваг власного продукту лише за появи сильного конкурента, перегляд виконання умов, що зменшують податки, поступове підвищення тарифів	T

На основі аналізу SWOT розроблено перелік заходів ринкової поведінки для виведення стартап-проекту на ринок. Орієнтовний оптимальний час їх ринкової реалізації з огляду на проекти конкурентів, що можуть бути виведені на ринок. Для визначених заходів виконано аналіз з точки зору строків та ймовірності отримання ресурсів для розробки мобільного додатку, результати якого наведенов Таблиці 7.13.

Таблиця 7.13

Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Створення ПЗ використовуючи патерн MVC	70%	12 місяців
2.	Створення ПЗ використовуючи патерн VIPER	55%	14 місяців

Аналізуючи результати з точки зору строків та ймовірності отримання ресурсів, наведеного у Таблиці 7.10, для реалізації проекту найкраже підходить альтернатива 1 - Створення ПЗ використовуючи патерн практичний Apple MVC, так як вона передбачає вищу ймовірність отримання ресурсів та більш стислі 8 строки реалізації, тобто є переважною по обом параметрам.

7.4 Розробка ринкової стратегії проекту

Розробка ринкової стратегії спочатку передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 7.14

Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Державні компанії	Спрощення процесу обміну даними	Середній	З конкурента з продуктами відповідної функціональності, проте без використання звукоих сигналів як спосіб обміну даними	Адаптивність, Новизна
2.	Приватні компанії	Спрощення процесу обміну даними	Великий		Адаптивність, Новизна
Обрано цільові групи: Державні компанії, Приватні компанії					

Результати розгляду цільових груп потенційних споживачів стартап-проекту, наведені в Таблиці 7.14, дозволяють визначити які переваги продукту можливо використати для виходу на цей сегмент ринку, а також чи є правильним витрата ресурсів для впливу на певну групу.

За результатами аналізу потенційних груп споживачів було обрано дві основні цільові групи, яким проект буде запропоновано для використання. Для ефективного впровадження продукту в обраних групах має бути розроблена стратегія охоплення ринку.

Таблиця 7.15

Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспромо жні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Створеня мобільного сервісу на основі паттерну MVC	Ринкове позиціонування	Адаптивність, Новизна, Простота у користуванні	Диференціація

За результатами визначення базової стратегії розвитку, представленими в Таблиці 7.15, обрано оптимальну стратегію розвитку за альтернативою, обраною в Таблиці 7.13, що має задовольняти основним потенційним групам споживачів стартап-проекту, розглянутим в Таблиці 7.14.

За допомогою результатів отриманих в ході вибору базової стратегії розвитку стартап-проекту, що в включає в себе стратегію охоплення ринку, альтернативу розвитку, конкурентноспроможні позиції, було обрано диференціальну стратегію. Також є необхідним і вибір стратегії конкурентної поведінки, наданий в Таблиці 7.16.

Таблиця 7.16

Визначення базової стратегії конкурентної поведінки

№ п/п	«Першопроходець» на ринку	Агресивний пошук нових споживачів	Копіювання основних характеристик в конкурент	Стратегія конкурентної поведінки
1.	Ні	Так	Так: - Адапртивність (к-ти 1 та 3) - простота використовування (к-т 1)	Зайняття конкурентної ніші

За результатами визначення базової конкурентної поведінки, представленими в Таблиці 7.16, було обрану оптимальну стратегію конкурентної поведінки за використання основних характеристик-переваг конкурентів, описаних в Таблиці 7.2, для якої були зазначені наявність агресивного пошуку споживачі та зайняття конкурентної ніші як стратегія

безпосередньо, що має надати можливість ефективної боротьби з вищезгаданими конкурентами.

На основі вимог споживачів з двох основних сегментів, описаних в Таблиці 7.14, до стартап-компанії та до її системи сервісів, зважаючи на диференціальну базову стратегіію розвитку та стратегії зайняття конкурентної ніші – це стратегії конкурентної поведінки, - було розроблено стратегію позиціонування, а саме сформовано ринкову позицію - комплекс асоціацій, для ідентифікації споживача торгівельної марки та проекту.

Таблиця 7.17

Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Адаптивність, простота в використанні, конфіденційність	Диференціація	Адаптивність, звук як спросід обміну даними, простота в використанні	Адаптивність Новизна Простота

За результатами представленими в Таблиці 7.17 робимо висновок. За орієнтації на основні вимоги до товару в цільової аудиторії було обрану оптимальну стратегію позиціонування стартап-проекту. З ключових конкурентоспроможних позицій стартап-проекту, наданих в Таблиці 7.15, в було обрано дві, що задовольняють поставленим вимогам: адаптивність та простота у використанні та реалізуванні. Третьою позицією для асоціацій було обрано не конкурентноспроможну позицію, а основну вимогу до будь-яких проектів в даній сфері - конфіденційність.

7.5 Розроблення маркетингової програми стартап-проекту

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у Таблиці 6.18 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 7.18

Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Економність в ресурсах мобільного додатку	Використання баз знань, що ефективно зберігають дані, використання звукових сигналів для обміну даними не використовуючи інтернет	Використання звукових сгналів для обміну даними
2.	Адаптивність	Включення нових модулів в систему без зайвих зусиль	Слабка зв’язаність компонентів

За результатами визначення ключових переваг концепції потенційного товару, представленими в Таблиці 7.18, було розглянуто основні потреби на ринку та проведено оцінку вигод проекту. Було визначено ключові переваги перед конкурентами, що задовольняють наведеним основним потребам. Такими ключовими перевагами стали використання баз знань та слабка зв’язаність компонентів (має бути впроваджено відповідно до патерну GRASP).

В Таблиці 7.19 наведено трирівневу маркетингову модель товару.

1-й рівень. Задум товару - засобом вирішення якої потреби / проблеми буде даний товар, яка його основна вигода (основа технічного завдання).

2-й рівень. Рішення реальної реалізації товару: якість, властивості, упаковка.

3-й рівень. Товар з підкріпленням (супроводом) - додаткові послуги та переваги для споживача, що створюються на основі товару за задумом і товару в реальному виконанні (гарантії якості , доставка, умови оплати та ін).

Таблиця 7.19

Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Мобільний додаток обміну захищеними даними з використанням звукових сигналів		
II. Товар у реальному виконанні	Властивості / характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Адаптивність	Нм	Технологічна
	2. Технологічність	Нм	Технологічна
	3. Конфіденційність	Нм	Технологічна
	4. Простота	Нм	Технологічна
	Якість: згідно до стандарту ISO 4444 буде проведено тестування		
	Маркування відсутнє		
	Моя компанія: “UniApp”		
III. Товар із підкріплюнням	Безкоштовна версія		
	Постійна підтримка для користувачів		
За рахунок чого потенційний товар буде захищено від копіювання: електронні ключі			

За результатами виконання трирівневої маркетингової моделі товару, представленими в Таблиці 7.19, було визначено задум товару його властивості та характеристики, стандарт якості, маркування та назва. Було визначено підкріплення товару – безкоштовна версія товару, регулярна підтримка.

Було визначено засоби захисту від копіювання: електронні ключі на фізичному рівні.

Надалі було визначено цінові межі, якими слід керуватись при встановленні ціни на потенційний товар, в ході чого було експертним методом

проведено аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів, результати якого представлені в Таблиці 7.20.

А також визначено оптимальну систему збуту, в межах якого приймається рішення, результати чого представлені в таблиці 7.21.

Таблиця 7.20

Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники, грн	Рівень цін на товари-аналоги, грн	Рівень доходів цільової групи споживачів, грн	Верхня та нижня межі встановлення ціни на товар/послугу, грн
1.	33000	45000	260000	27000-40000

Таблиця 7.21

Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Придбання місячної або річної підписки	Продаж	0 (напряму) 1 (через одного посередника)	Власна та через посередників

За результатами визначення цінових меж та оптимальної системи збуту, представленими в Таблиці 7.20 та Таблиці 7.21, із врахуванням цін на товари-аналоги та товари-замінники було встановлено верхню та нижню границі ціни. До того ж було сформовано систему збуту, що передбачає продовження ліцензії шляхом придбання періодичної підписки: місячної чи річної (дешевшої за місячну сумарно) та описано канали збуту, в результаті чого система збуту була визначена як власна та через посередників.

В якості останньої складової маркетингової програми було розроблено концепцію маркетингових комунікацій, для чого було використано результат попереднього вибору основи для позиціонування та визначену специфіку поведінки клієнтів, результати розробки якої наведено в Таблиці 7.22.

Таблиця 7.22

Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Придбання повної версії шляхом отримання унікального ключа	Інтернет	Адаптивність Конфіденційність Простота	Показати переваги сервісу, у тому числі і перед конкурентами	Демо-ролик застосування Таргетингова реклама

За результатами розробки концепції маркетингових комунікацій, представленими в Таблиці 7.22, із використанням результатів формування системи збуту, наведених в Таблиці 7.21, було визначено специфіку поведінки цільових клієнтів; Інтернет, як головний канал комунікації цільових клієнтів; Адаптивність, Конфіденційність, Простота як ключові позиції для позиціонування; а також сформовані завдання рекламного повідомлення, концепція рекламного звернення.

Висновок по розділу

В ході складання документації було сформовано ідею стартап-проекту “Мобільний додаток обміну захищеними даними з використанням звукових сигналів ”, наведено порівняння із рядом потенційних конкурентів, що надало можливість оцінки конкурентоспроможності та можливості і складності виходу стартапу на ринок. Також було розглянуто основні технології, що можуть бути використані з метою реалізації системи стартап-проекту. Надано результати аналізу характеристик попиту на ринку. Був проведений аналіз цільових аудиторій споживачів описаного продукту, також були розглянуті можливі фактори загроз а також сприятливих умов для стартап-проекту.

Проаналізувавши та обґрунтувавши фактори конкурентоспроможності, та розробивши порівняльний аналіз сильних та слабких сторін проекту, результати були використані для SWOT-аналізу стартап-проекту.

Було обрано цільові групи потенційних споживачів, також визначено базову стратегію розвитку стартап-проету та стратегії конкурентної поведінки та позиціонування. Розроблено маркетингову програму.

Тому для успішного виконання проекту необхідно реалізувати алгоритм роботи сервісів Firebase разом з мобільним додатком який базується на патерні «Практичний Apple MVC». В рамках цього розділу були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків.

На основі цих результатів робимо висновок про те, що подальша імплементаціє є доцільною.

ВИСНОВОК

Метою роботи була розробка мобільного додатку, який використовує звукові сигнали як спосіб обміну даними між користувачами, а також й інші методи за допомогою яких можна здійснювати обмін даними.

У ході роботи було проаналізовано дві мови програмування Objective C та Swift. Позитивні та негативні сторони для подальшого використання при розробці мобільного додатку.

Також важливою частиною дисертації є аналіз патернів програмування, на цьому етапі роботи було розглянуто сім архітектурних шаблонів, та було обрано один для створення архітектури додатку.

В дисертації наведено аналіз трьох технологій обміну даними. Розглянуті сильні та слабкі сторони технологій для визначення пріоритетів при розробці мобільного додатку.

Для захисту даних при обміні були розглянуті стандарти шифрування та обрано той стандарт який є більш доцільним для використання завдяки їх аналізу.

Тому реалізований мобільний додаток є унікальним серед існуючих аналогів. Визначена платформа розробки - Objective-C, середовище розробки Xcode, архітектура мобільного додатку - практичний Apple MVC. Технології обміну даними є інтернет, QR-код, фреймворк «Chirp Connect» для використання звукових сигналів. Стандарт шифрування даних – AES128. Були вивчені основні принципи дизайну інтерфейсу для мобільних пристроїв, які будуть використовуватися для подальшої функціональної розробки додатку.

Розроблено призначений для користувача інтерфейс програми, максимальне число переходів між екранами для виконання тестових сценаріїв дорівнює вісім, що дозволяє користувачу швидко здійснювати свою діяльність з додатком. Базова функціональність програмного забезпечення, реалізована в рамках магістерської дисертації, і використана архітектура передбачає подальше нарощування його функціональних можливостей.

Список використаних джерел

1. Objective-C - Вікіпедія [Електронний ресурс] / Wikimedia Foundation, Inc. - Режим доступу: URL: https://uk.wikipedia.org/wiki/Objective-C. - Загл. з екрану. - яз. рус., англ.
2. Swift_(язык_программирования) - Вікіпедія [Електронний ресурс] / Wikimedia Foundation, Inc. - Режим доступу: URL: https://ru.wikipedia.org/wiki/Swift_(язык_программирования). - Загл. з екрану. - яз. рус., англ.
3. Swift - [Електронний ресурс] / Apple Inc. - Режим доступу: URL: https://www.apple.com/swift/. - Загл. з екрану. - яз. англ.
4. Язык программирования Swift. Русская версия - [Електронний ресурс] . - Режим доступу: URL: https://habr.com/post/225841/. - Загл. з екрану. - яз. рус.
5. Інтернет - Вікіпедія [Електронний ресурс] / Wikimedia Foundation, Inc. - Режим доступу: URL: https://uk.wikipedia.org/wiki/Інтернет. - Загл. з екрану. - яз. укр., англ.
6. Під'єднання та обмін даними в Інтернеті - [Електронний ресурс]. Режим доступу: URL: https://mozok.click/644-pdyednannya-ta-obmn-danimi-v-nternet.html. - Загл. з екрану. - яз. укр.
7. QR-код - Вікіпедія [Електронний ресурс] / Wikimedia Foundation, Inc. - Режим доступу: URL: https://ru.wikipedia.org/wiki/QR-код. - Загл. з екрану. - яз. рус.
8. Батрак В.І. Використання qr кодів [Електронний документ]. – Режим доступу: URL: https://www.google.com.ua/url?sa=t&rct=j&q=&esrc=s&source=web&cd=15&ved=2ahUKEwinn5XV7o3fAhUulIsKHWeLCxMQFjAOegQIARAC&url=http%3A%2F%2Fe-journals.npu.edu.ua%2Findex.php%2Fikt%2Farticle%2Fdownload%2F20%2Fpdf_18&usg=AOvVaw2kEhetIXX9ZRyasIQsgUXc

9. Chirp Connect Overview [Електронний ресурс]. – Режим доступу: URL: https://chirp.io. - Загл. з екрану. – яз. англ.
10. Шаблон проектирования MVC [Електронний ресурс]. - Режим доступу: URL: https://webformyself.com/shablon-proektirovaniya-mvc/ . - Загл. з екрану. - яз. рус., англ.
11. Model-View-Controller (MVC) in IOS: A Modern Approach [Електронний ресурс]. - Режим доступу: URL: https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach . - Загл. З екрану. - яз. англ.
12. IOS Architecture Patterns [Електронний ресурс]. - Режим доступу: URL: https://techblog.badoo.com/blog/2016/03/21/ios-architecture-patterns/ . - Загл. З екрану. - яз. англ.
13. Реализация MVVM в IOS с помощью RxSwift [Електронний ресурс]. - Режим доступу: URL: https://habrahabr.ru/post/273455/ - Загл. з екрану. - яз. рус., англ.
14. The Book of VIPER [Електронний ресурс]. - Режим доступу: URL: https://github.com/strongself/The-Book-of-VIPER . - Загл. з екрану. - яз. англ.
15. XCode - IDE - Apple Developer [Електронний ресурс] / Apple Inc. - Режим доступу: URL: https://developer.apple.com/xcode/ide/. - Загл. з екрану. - яз. англ.
16. Core Data Programming Guide: Persistent Store Types and Behaviors [Електронний ресурс] / Apple Inc. - Режим доступу: URL: https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual / CoreData /. - Загл. з екрану. - яз. англ.
17. XMPP is the open standard for messaging and presence [Електронний ресурс] / XMPP Community. - Режим доступу: URL: http://xmpp.org/. - Загл. з екрану. - яз. англ.
18. Push-повідомлення - Вікіпедія [Електронний ресурс] / Wikimedia Foundation, Inc. - Режим доступу: URL:

http://ru.wikipedia.org/wiki/Push_уведомления. - Загл. з екрану. - яз. рус., англ.

19. Стандарт, ГОСТ 28147-89. Система обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. [Електронний документ]. - Режим доступу: URL: http://protect.gost.ru/document.aspx?control=7&id=139177
20. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. /Б.Шнайер/ [Текст] –М.: Издательство ТРИУМФ, 2002 – 816 с.
21. А.Ализар, Представлен первый рабочий метод снижения криптостойкости алгоритма AES, [Електронний документ]. - Режим доступу: URL: https://xakep.ru/2011/08/18/56520/
22. Пржиялковский В.Oracle10: шифруем данные, [Електронний документ]. - Режим доступу: URL: http://citforum.ru/database/oracle/crypt_data/
23. Основные параметры. Rijndael (AES). [Електронний документ] . - Режим доступу: URL: http://www.enlight.ru/crypto/algorithms/rijndael/rijndael00.htm
24. Сравнение настольных программ для шифрования [Електронний документ] . - Режим доступу: URL: https://habrahabr.ru/company/cybersafe/blog/252561/
25. Шифрование XTS [Електронний документ] . - Режим доступу: URL: http://www.kingston.com/ru/usb/encrypted_security/xts_encryption
26. Сравнение средств шифрования [Електронний документ] . - Режим доступу: URL: http://efsol.ru/articles/comparison-encryption.html
27. Цибульов П. М. Управління інтелектуальною власністю : монографія/ Цибульов П. М., Чеботарьов В. П., Зінов В. Г. , Суїні Ю., за ред. П. М. Цибульова. – К. : «К. І. С.», 2005. – 448 с.
28. Квашнин А. Как управлять портфелем технологий и интеллектуальной собственностью : серия методических материалов «Практические руководства для центров коммерциализации технологий» / под рук. П.

Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 60 с.

29. Квашнин А. Как продвигать проекты коммерциализации технологий : серия методических материалов «Практические руководства для центров коммерциализации технологий» / М. Катешова, А. Квашнин, под рук. П. Линдхольма, проект EuropeAid «Наука и коммерциализация технологий», 2006. – 52 с.
30. Петруненков А. Оценка коммерческой привлекательности проекта [Електронний ресурс] // Технологический бизнес. – 1999. – № 2. Режим доступу: URL: http://www.techbusiness.ru/tb/archiv/number2/ page01.htm
31. Бланк, С. Стартап. Настольная книга основателя / С. Бланк, Б. Дорф ; пер. с англ. Т. Гутман, И. Окунькова, Е. Бакушева. – 2-е изд. – Москва : Альпина Паблишер, 2014. – 614 с.
32. Дрейпер, У. Стартапы : профессиональные игры Кремниевой долины / У. Дрейпер ; предисл. Э. Шмидта ; пер. с англ. В. Егорова. – Москва : Эксмо, 2012. – 378 с.
33. Коэн, Д. Стартап в Сети : мастер-классы успешных предпринимателей / Д. Коэн, Б Фелд ; пер. с англ. М. Иутина. – 2-е изд. – Москва : Альпина Паблишер, 2013. – 337 с.
34. Маллинс, Дж. Поиск бизнес-модели : как спасти стартап, вовремя сменив план / Дж. Маллинс, Р. Комисар ; пер. с англ. М. Пуксант и Е. Бакушевой. – Москва : Манн, Иванов и Фербер, 2012. – 329 с.
35. Робемед, Н. Самые интересные стартапы 2013 года [Електронний ресурс]. – Режим доступу: URL: http://www.forbes.ru/svoi-biznes-photogallery/ startapy/248976-samye-interesnye-startapy-2013-goda/photo/1
36. Статистика смертности и советы по безопасности для стартапов [Електронний ресурс]. – Режим доступу: URL: https://vc.ru/p/startup-eset
37. Статистика указала на условия для появления стартапов, успешных как Google и Facebook [Електронний ресурс]. – Режим доступу: URL: https://naked-science.ru/article/sci/statistika-ukazala-na-usloviya

38. Тиль, П. От нуля к единице : как создать стартап, который изменит будущее / П. Тиль, Б. Мастерс; перевод с англ. – Москва : Альпина паблишер, 2015. – 188 с.
39. Харниш, В. Правила прибыльных стартапов : как расти и зарабатывать деньги / В. Харниш ; пер. с англ. В. Хозинского. – Москва : Манн, Иванов и Фербер, 2012. – 279 с.
40. Экланд С. Ангелы, драконы и стервятники : как привлечь правильных инвесторов в свой стартап и сохранить бизнес / С. Экланд ; пер. с англ. О. Терентьевой. – Москва : Манн, Иванов и Фербер, 2011. – 275 с.
41. Jurg van Vliet, Programming Amazon EC2, O'Reilly Media, 2011, 186c
42. Jurg van Vliet, Flavia Paganelli, Jasper Geurtsen, Resilience and Reliability on AWS, O'Reilly Media, 2013, 164c
43. Amazon Web Services, Amazon Elastic Compute Cloud (EC2) User Guide, Amazon Digital Services, Inc., 2012, 892c
44. Rob Linton, Amazon Web Services: Migrating your .NET Enterprise Application, Packt Publishing, 2011, 311c
45. James Murty, Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB, O'Reilly Media, 2009, 604c
46. Danny Garber, Jamal Malik, Adam Fazio, Windows Azure Hybrid Cloud, Wrox, 2013, 246c
47. Tejaswi Redkar, Tony Guidici, Windows Azure Platform, Apress, 2011, 602c
48. Sriram Krishnan, Programming Windows Azure, O'Reilly Media, 2010, 370c
49. Roger Jennings, Cloud Computing with the Windows Azure Platform, Wrox, 2010, 360c
50. Firebase - Вікіпедія [Електронний ресурс] / Wikimedia Foundation, Inc. - Режим доступу: URL: https://uk.wikipedia.org/wiki/Firebase . - Загл. з екрану. - яз. укр., англ.

ДОДАТОК А

Основні цінові параметрами сервісу Firebase

Products	Spark Plan Generous limits for hobbyists Free	Flame Plan Fixed pricing for growing apps $25/month	Blaze Plan Calculate pricing for apps at scale Pay as you go ✓ Free usage from Spark plan included*
Free Products A/B Testing, Analytics, App Indexing, Authentication (except Phone Auth), Cloud Messaging (FCM), Crashlytics, Dynamic Links, Invites, Performance Monitoring, Predictions, and Remote Config.	✓ Included	✓ Included Free	✓ Included Free
Realtime Database			
Simultaneous connections	100	100k	100k/database
GB stored	1 GB	2.5 GB	$5/GB
GB downloaded	10 GB/month	20 GB/month	$1/GB
Multiple databases per project	✗	✗	✓
Cloud Firestore			
Stored data	1 GB total	2.5 GB total	$0.18/GB
Bandwidth	10GB/month	20GB/month	Google Cloud Pricing
Document writes	20K/day	100K/day	$0.18/100K
Document reads	50K/day	250K/day	$0.06/100K
Document deletes	20K/day	100K/day	$0.02/100K
Storage			
GB stored	5 GB	50 GB	$0.026/GB
GB downloaded	1 GB/day	50 GB/day	$0.12/GB
Upload operations	20K/day	100K/day	$0.05/10k
Download operations	50K/day	250K/day	$0.004/10k
Multiple buckets per project	✗	✗	✓
Cloud Functions			
Invocations	125K/month	2M/month	$0.40/million
GB-seconds	40K/month	400K/month	$0.0025/thousand
CPU-seconds	40K/month	200K/month	$0.01/thousand
Outbound networking	Google services only	5 GB/month	$0.12/GB
Phone Auth			
US, Canada, and India	10k/month	10k/month	$0.01/verification
All other countries	10k/month	10k/month	$0.06/verification
Hosting			
GB stored	1 GB	10 GB	$0.026/GB
GB transferred	10 GB/month	50 GB/month	$0.15/GB
Custom domain & SSL	✓	✓	✓
Multiple sites per project	✗	✗	✓
Test Lab			
Virtual Device Tests	10 tests/day	10 tests/day	$1/device/hour
Physical Device Tests	5 tests/day	5 tests/day	$5/device/hour
ML Kit			
On-Device APIs	✓	✓	✓
Custom Model Hosting/Serving	✓	✓	✓
Cloud Vision APIs	✗	✗	$1.50/K
Google Cloud Platform			
Use BigQuery & other IaaS	✗	✗	✓

Рис. А.1 – Основні цінові параметрами користування сервісом Firebase

ДОДАТОК Б

Алгоритм роботи мобільного додатку з під час пошуку нових користувачів

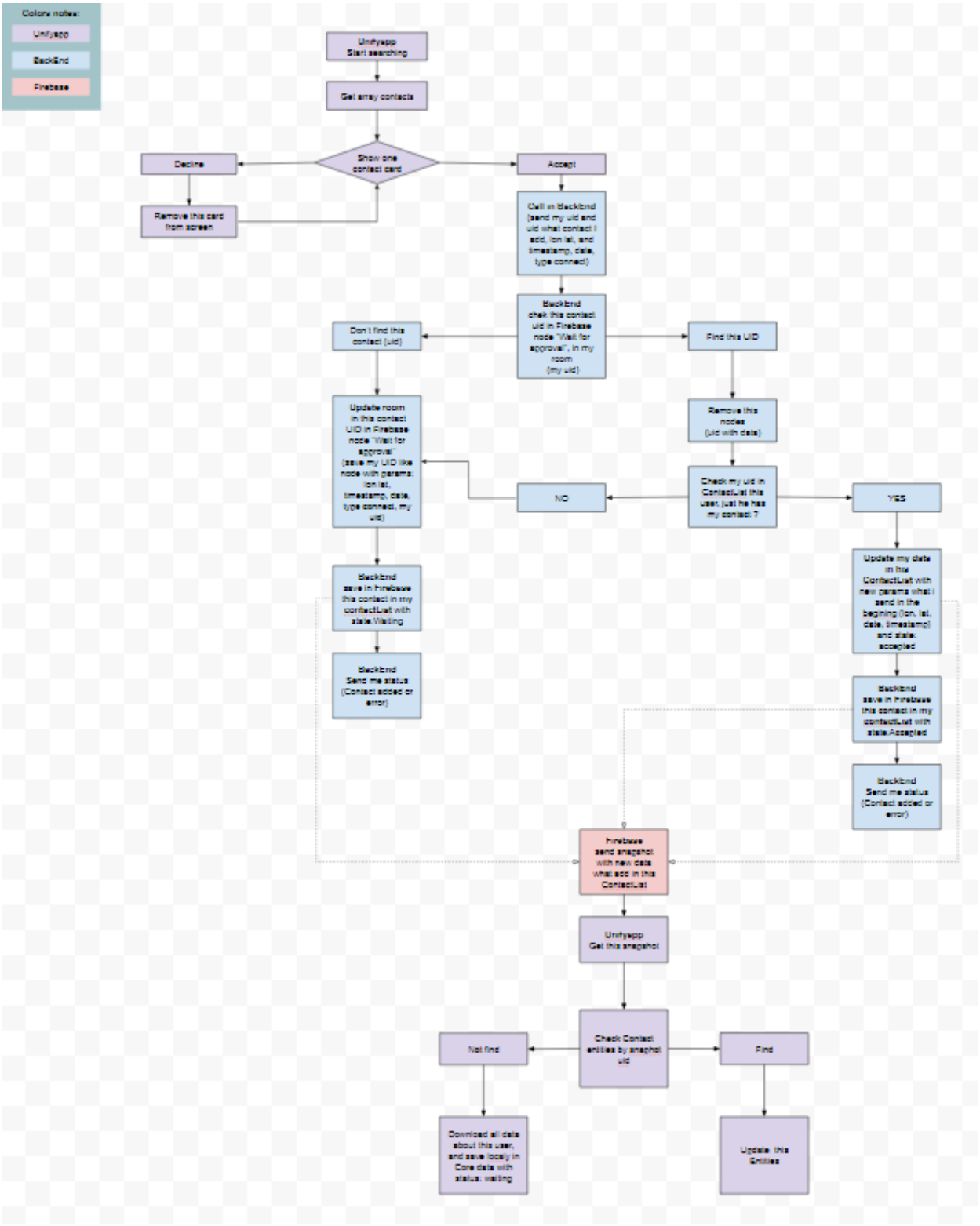


Рис. Б.1 – Алгоритм пошуку нових контактів через Firebase

ДОДАТОК В

Алгоритм роботи QR-коду

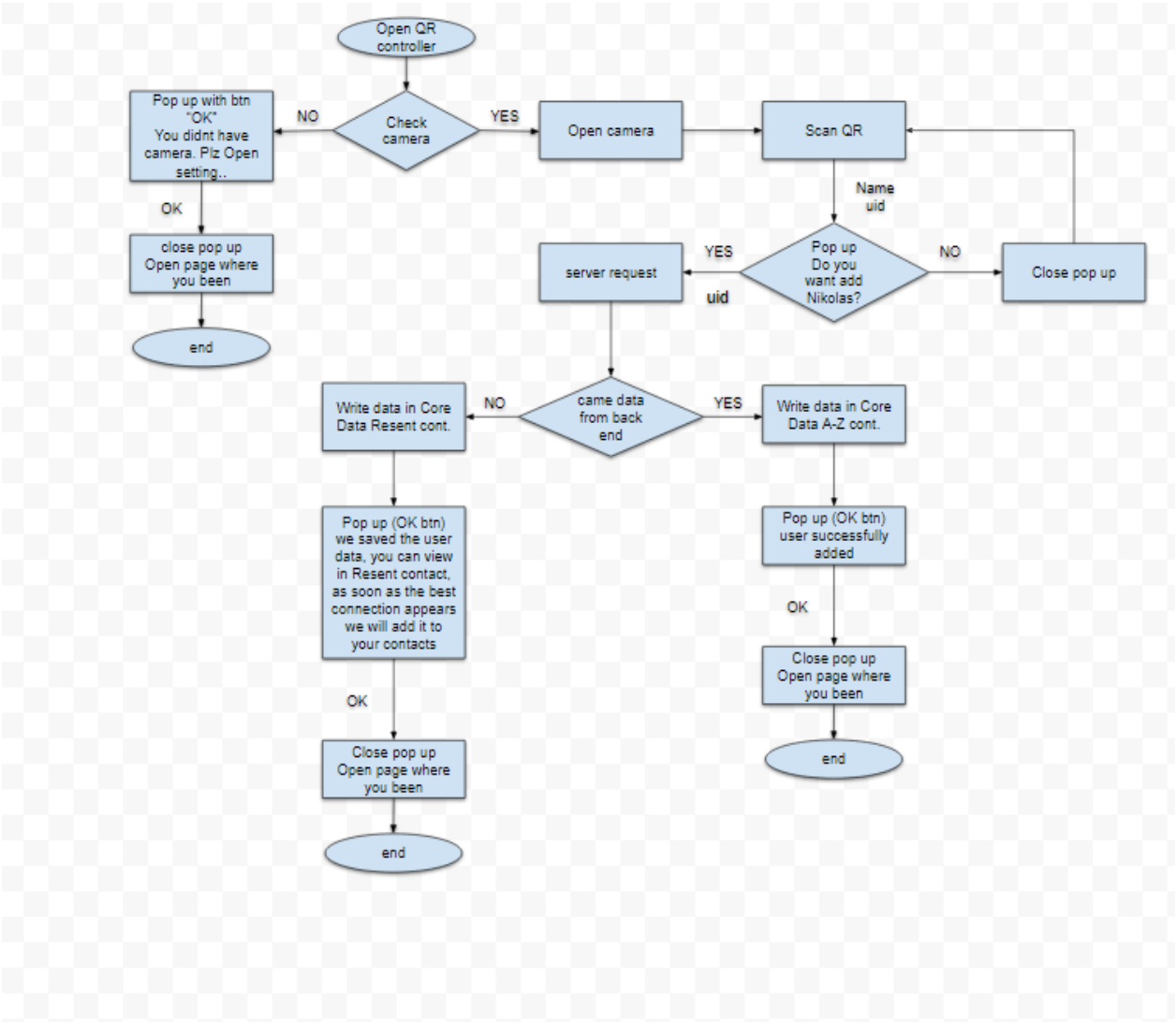
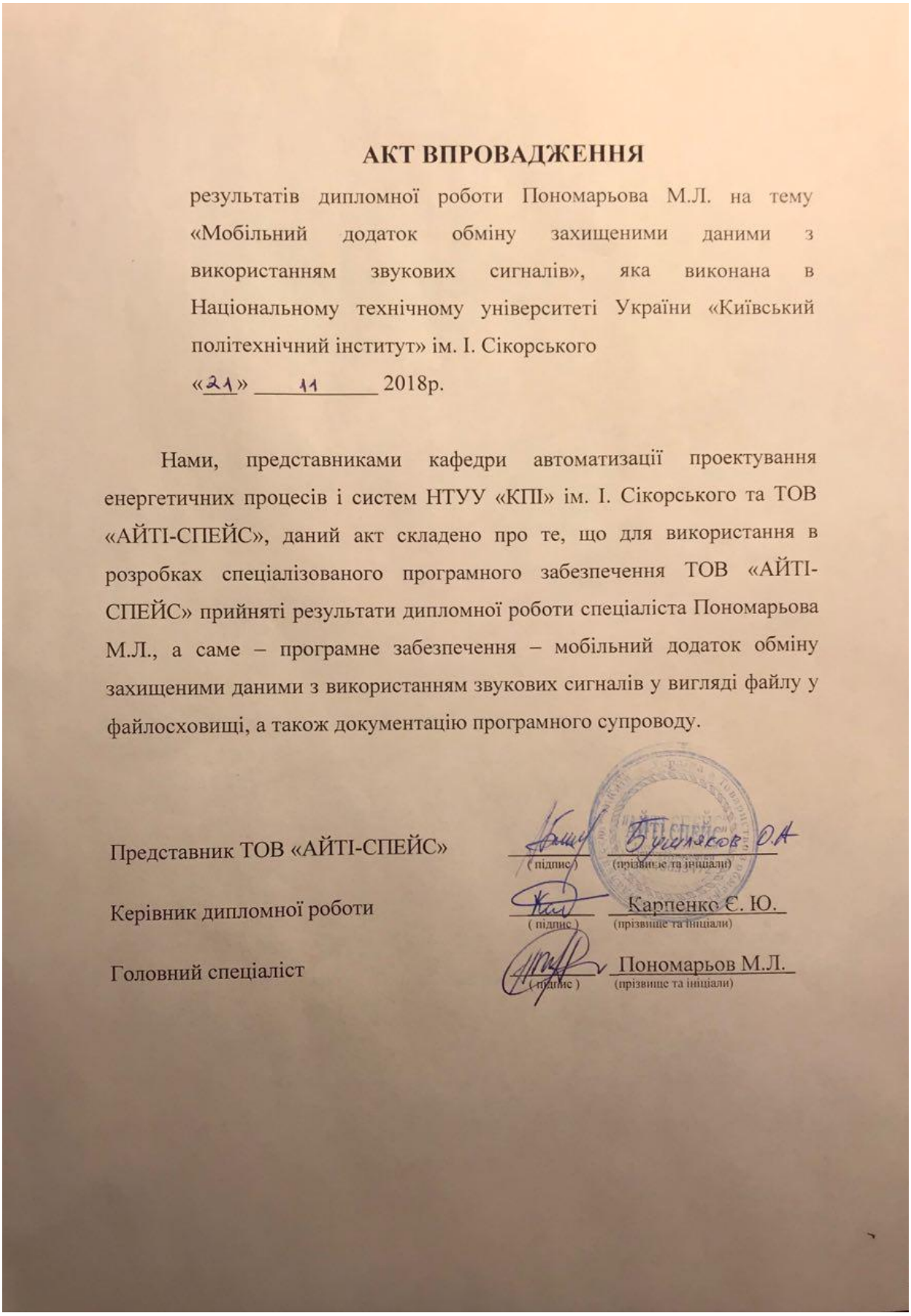


Рис. В.1 – Алгоритм роботи QR-коду при додаванні нового контакту

ДОДАТОК Д

Фото акту впровадження



АКТ ВПРОВАДЖЕННЯ

результатів дипломної роботи Пономарьова М.Л. на тему «Мобільний додаток обміну захищеними даними з використанням звукових сигналів», яка виконана в Національному технічному університеті України «Київський політехнічний інститут» ім. І. Сікорського

«21» 11 2018р.

Нами, представниками кафедри автоматизації проектування енергетичних процесів і систем НТУУ «КПІ» ім. І. Сікорського та ТОВ «АЙТІ-СПЕЙС», даний акт складено про те, що для використання в розробках спеціалізованого програмного забезпечення ТОВ «АЙТІ-СПЕЙС» прийняті результати дипломної роботи спеціаліста Пономарьова М.Л., а саме – програмне забезпечення – мобільний додаток обміну захищеними даними з використанням звукових сигналів у вигляді файлу у файлосховищі, а також документацію програмного супроводу.

Представник ТОВ «АЙТІ-СПЕЙС» _____ (підпис) Буцяков О.А (прізвище та ініціали)

Керівник дипломної роботи _____ (підпис) Карпенко Є. Ю. (прізвище та ініціали)

Головний спеціаліст _____ (підпис) Пономарьов М.Л. (прізвище та ініціали)

ДОДАТОК Е

Тези на тему «Мобільний додаток обміну захищеними даними з використанням звукових сигналів»

МОБІЛЬНИЙ ДОДАТОК ОБМІНУ ЗАХИЩЕНИМИ ДАНИМИ З ВИКОРИСТАННЯМ ЗВУКОВИХ СИГНАЛІВ

Пономарьов Микола Леонідович
Науковий керівник: к.т.н, доцент Карпенко Є.Ю.
Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського", Україна

Мобільні додатки стали одним з головних трендів у розвитку інформаційних технологій в останні роки. Основною перевагою мобільних додатків є те, що користувач може отримати доступ до них, перебуваючи в будь-якому місці та в будь-якій ситуації. Тому важливу роль відіграє чи може користувач оперувати та ділитися даними зі свого пристрою.

Наприклад, одним з безлічі типів додатків для мобільних пристроїв є додатки для подорожей і туризму. Під час таких подорожей вони повинні краще працювати в офлайн-режимі, тому що то не завжди у будь-якому місці та в будь-якій ситуації, у вас буде доступ до інтернету або мобільного оператора.

На основі аналізу технологій обміну даними що існують, був розроблений мобільний додаток з архітектурою, яка зображена на рис. 1.

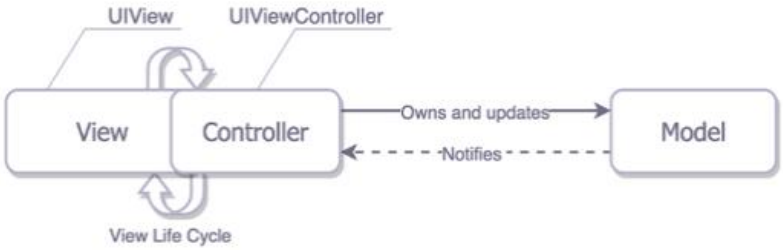


Рис. 1 – Практичний Apple MVC

У цьому шаблоні архітектури, контролер буде посередником між View і Model, що б ці два об'єкти не знали про існування один одного [1].

У мобільному додатку було обрано три технології для обміну даними: інтернет, QR-код, звукові сигнали. Внаслідок використання звукових сигналів пристрій стає більш автономним для обміну даними, тому що не всі мобільні пристрої мають камеру, або належного операційного обладнання для зчитування QR-коду.

Список використаних джерел

1. Model-View-Controller (MVC) in IOS: A Modern Approach [Електронний ресурс] / raywenderlich.com: [сайт]. – Режим доступу: https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach, вільний. – Назва з екрану.

ДОДАТОК Є

Результати перевірки дисертації на плагіат